System Administration

Varian NMR Spectrometer Systems
With VNMR 6.1C Software

Pub. No. 01-999166-00, Rev. C0503

VARIAN

System Administration
Varian NMR Spectrometer Systems
With VNMR 6.1C Software
Pub. No. 01-999166-00, Rev. C0503

Revision history:
A0800 – Initial release for VNMR 6.1C
A1001 – Corrected errors on pg 120, general edit
B0202 – Updated AutoTest
B0602 – Added additional Autotest sections including VNMRJ update
B1002 – Updated Solaris patch information and revised section 21.7, Autotest
C0503 – Add additional Autotest sections including cryogenic probes

Applicability:
Varian NMR spectrometer systems with Sun workstations
running Solaris 2.x and VNMR 6.1C software

By Rolf Kyburz (rolf.kyburz@varianinc.com)
Varian International AG, Zug, Switzerland,
and
Gerald Simon (gerald.simon@de.varioan.com)
Varian GmbH, Darmstadt, Germany

Additional contributions by Frits Vosman, Dan Iverson, Evan Williams, George Gray, Steve Cheatham
Technical writer: Mike Miller
Technical editor: Dan Steele

Overview of Contents

SAFETY PRECAUTIONS 17
Disclaimer 21
Foreword 21
Acknowledgments 21
Conventions in This Manual 22
Chapter 1. Sun Hardware 23
Chapter 2. Software Installation 39
Chapter 3. VNMR-Related System Administration 61
Chapter 4. Output Devices, Terminals, Serial Ports 85
Chapter 5. Software Operating Levels 93
Chapter 6. Disk Organization 105
Chapter 7. Files, Permissions, and Owners 119
Chapter 8. UNIX Commands 131
Chapter 9. Text Editing 141
Chapter 10. Using Tapes and Floppy Disks 149
Chapter 11. Periodic Maintenance and Troubleshooting 163
Chapter 12. File Security, Repair, and Archiving 175
Chapter 13. UNIX System Customization 183
Chapter 14. A Quick Guide to C 189
Chapter 15. Shell Scripts 205
Chapter 16. Tcl/Tk 219
Chapter 17. Networking via Ethernet 221
Chapter 18. X Window System Operation 249
Chapter 19. Other Networking Options 261
Chapter 20. Solaris 2.x vs. SunOS 4.1.x 267
Chapter 21. Calibration Tests and Shimming 269
Chapter 22. Magnet and Spectrometer Maintenance 317
Annotated Bibliography 329
Glossary of UNIX, Sun, and VNMR Terms 337
Index 397

Table of Contents

SAFETY PRECAUTIONS 17

Disclaimer 21

Foreword 21

Acknowledgments 21

Conventions in This Manual 22

Chapter 1. Sun Hardware 23

1.1 How Does a Workstation Work? 23

- CPU Module 24
- CPU Chip 24
- Random-Access Memory 25
- Graphics Controller 25
- System Interconnect 26
- PCI Bus / SBus 26
- EBus 27
- Other CPU Board Components 27
- SCSI Bus 27

1.2 Sun Workstations—An Overview 28

- CPU Type, Performance 29
- Expansion Options, I/O Ports. 29

1.3 Data Storage Options 29

- Hard Disk Drives 29
- Floppy Disk Drives 31
- CD-ROM Drives 32
- Tape Drives 33
- Read/Write Optical Disk Drives, WORM Drives 35

1.4 Installing the Hardware 35

1.5 Adding a Disk (Hardware) 35

- Adding an Internal Disk 35
- Adding an External Hard Disk 36
- Checking the SCSI Address 36
- Hints on Adding SCSI Devices 37

Chapter 2. Software Installation 39

2.1 Installing Solaris 39

- Swap Space 40
- Planning the Disk Layout for Solaris 2.5.1 40
- Planning the Disk Layout for Solaris 2.6 43
- Adjusting the Disk Layout, Installing Solaris 2.5.1 44
- Solaris Patch Installation 48
- Assigning a Password to the System Administrator 48

2.2 Installation and Customization of VNMR 49
Installing New Users 49
Assigning New Users 49
Sending System Messages 51
2.3 Adding a Disk (Software) 51
2.4 Adding Memory 55
2.5 Increasing Swap Space 56
2.6 Changing the Hostname 57
2.7 Adding in Parts of Solaris “After the Fact” 58

Chapter 3. VNMR-Related System Administration 61
3.1 Acquisition Processes 61
UNITY*INOVA* and *MERCURY-VX* Systems 61
MERCURY and GEMINI 2000 Systems 62
UNITY*plus*, UNITY, and VXR-S Systems 63
3.2 Stopping and Restarting Acquisition Processes 65
Stopping Acqproc 65
Starting Acqproc 66
Rebooting the Acquisition System 67
Controlling the Acquisition Process in Multiuser Environments 67
3.3 Shutting Down and Restarting UNIX 68
Normal UNIX Shut Down 68
Shutting Down UNIX in an Emergency 69
Restarting UNIX 69
3.4 Configuring the VNMR User’s Environment 71
Adding a User 71
Resizing Windows 71
3.5 Shutting Down and Restarting the Host Computer 71
3.6 Modifying the Nucleus Tables 72
3.7 Changing the Standard Parameter Sets 73
3.8 User-Selected Shim Buttons 73
3.9 File Protection 74
Hiding a Command 75
3.10 VNMR Accounting Tool 76
Tutorial: Using the Accounting Tool 76
Navigating the VNMR Accounting Tool 84

Chapter 4. Output Devices, Terminals, Serial Ports 85
4.1 /vnmr/devicenames File 85
4.2 /vnmr/devicetable File 86
4.3 Printing and Plotting under VNMR 87
Printing and Plotting Files 87
vnmrprint Shell Script 87
vnmrplot Shell Script 88
4.4 UNIX Aspects of Printing and Plotting 88
How to Use Printers Under Solaris 89
How Does Solaris Printing Work? 89
4.5 Null Modems for Printers and Plotters 89
4.6 Fixing Problems with Printing and Plotting 90

4.7 Connecting Serial Terminals 90
4.8 Using Extra Ports 91
4.9 Using an I/O Port for Acquisition Diagnostics 91

Chapter 5. Software Operating Levels 93

5.1 Solaris Run Levels 93
Run Level 0—Monitor Mode 93
Run Level 1—Single-User Mode 95
Run Level 2—UNIX Without Networking 95
Run Level 3—Networking Included 96
init and shutdown Commands 96
How Does the UNIX Bootup and Shutdown Work? 97
5.2 Login Shell 99
5.3 Windowless Multiuser Mode 100
5.4 Graphical User Interface 100
Common Desktop Environment 101
OpenWindows 102
Shell Tool, Command Tool, Dtterm, and Xterm 102
VNMR 103
5.5 Shutting Down the System 103
To Shut Down the Entire System 104
To Shut Down Acquisition Computer or Acquisition Cabinet 104

Chapter 6. Disk Organization 105

6.1 Disk Partitioning 105
Naming Disk Slices 105
A Typical Solaris System Disk Layout 107
How is a Disk Organized Internally? 107
ufs File System 108
6.2 Solaris Directory Structure 109
Files in the Root Slice 110
/tmp Slice 112
/var Slice 113
/opt Slice 113
/usr File System 114
Home Directory Slice, /export/home 115
Files on Other Disks or Disk Slices 117
6.3 Multiuser Setup for VNMR 117

Chapter 7. Files, Permissions, and Owners 119

7.1 How to Get File Information 119
7.2 Protection Bits 119
7.3 Owners and Groups 121
Changing File Ownership 121
Changing Group Ownership 121
7.4 Special Permissions 122
7.5 Hard Links, Symbolic Links: the UNIX File System 124
File System Tree 124
File Storage 124
Directory Files 126

Hard Links 127
Symbolic Links 128

Chapter 8. UNIX Commands 131

8.1 Command Syntax 131
Command Arguments 131
Online Command Reference 132
Command Grouping 132
Background Execution 132
Input and Output Redirection 132

8.2 File Names 133
File Name Suffix 133
Dot Files 133
File System Hierarchy 133
Wildcard Characters 134
Keyboard Shortcuts for Entering File Names 134

8.3 Important UNIX Commands 135
Directory Commands 135
File Handling Commands 135
Text Commands 137
Process Management and System Administration Commands 137

8.4 Interactive Work with the C Shell 138

8.5 Shortcuts within Open Look and CDE 139

Chapter 9. Text Editing 141

9.1 Text Editors dtpad and textedit 141

9.2 Text Editor vi 142
Command Mode 142
Insert Mode 144
Last Line Mode 144
Other vi commands 145
Problems with Running vi 145

9.3 Other Editors 147
ed and ex Editors 147
emacs Editor 147

Chapter 10. Using Tapes and Floppy Disks 149

10.1 tar Command 149
Command Syntax 149
tar Files 150
QIC Tape Formats 151
Blocking Factor 151
Writing Files 151
Tape Catalogs 152
Reading Files 152
Verbose Option 153
Making Things Simpler 153
Using tar to Transfer Directories 153
Using Multifile Tapes 154
Using tar Remotely 154

10.2 ufsdump, ufsrestore, and dd Commands 155
Reading Gemini and VXR-4000 Tapes 155
10.3 Using Floppy Disks 156
Formatting Floppy Disks 156
Floppies with the tar File System 157
Floppies with DOS Format 157
Floppies with UNIX File System 158
Transferring Large Files Via Floppy Disks 160
Reading bar Floppies 161

Chapter 11. Periodic Maintenance and Troubleshooting 163
11.1 Daily and Periodical Care of Computer Hardware 163
11.2 Managing Free Space on Disks 163
11.3 Automatic File System Maintenance 164
11.4 Troubleshooting 165
Acquisition Status Codes 165
Converting References to Subnet 10.0.0 (UNITY*INOVA*) 167
Display Hardware Console Status (UNITY*INOVA)* 168
Global Parameter File Corrupted 168
Insufficient Swap Space 168
tmp Partition Fills Up 169
Password Invalid 169
System Hangs 170
SCSI Communications Unreliable 170
Changing the Name of a Solaris Host 170
Ethernet Network Troubleshooting 171
System Failure 172

Chapter 12. File Security, Repair, and Archiving 175
12.1 fsck Command 175
12.2 ufsdump Command 176
12.3 ufsrestore Command 178
12.4 tar Backups 179
12.5 Backups Using cron 179
12.6 Data Compression 179
What Files Can Be Compressed? 180
Procedures for Compressed Archiving 180

Chapter 13. UNIX System Customization 183
13.1 General UNIX Customization 183
.login File 183
.cshrc File 184
.exrc File 185
.profile File 185
.mailrc File 185
.indent.pro File 185
13.2 X Window System Configuration Files 186
.xinitrc File 186
.Xdefaults File 186
.xlogin File 186

13.3 CDE Customization Files 186
13.4 OpenWindows Configuration Files 186
.openwin-init File 186
.openwin-menu File 187
Other Customization Files 187
13.5 VNMR and Other Application Configuration Files 187

Chapter 14. A Quick Guide to C 189
14.1 General Program Structure 190
14.2 Constants, Preprocessor Functions 191
14.3 Types 192
14.4 Variables 193
14.5 Operators 195
14.6 Flow Control 197
Branching 197
Looping 198
14.7 Functions 199
14.8 Pointers 200
14.9 Input/Output 201
14.10 Syntax Checking and Compilation 203
14.11 How Should C Programs Be Formatted? 204

Chapter 15. Shell Scripts 205
15.1 C Shell Scripts 205
C Shell Selection, Comments 205
Set-Variables 206
Environment Variables 206
Arithmetics with Variables 207
Flow Control 207
Arguments 210
Interactive Input 211
Here Documents 211
15.2 Bourne Shell Scripts 212
Bourne Selection, Comments 212
Variables 212
Environment Variables 212
Arithmetic with Variables 212
Flow Control 213
Conditional Execution 214
Looping 215
`case` Construction 215
Arguments 216
Interactive Input 217
Here Documents 217
Procedures 218
15.3 How to Write Shell Scripts 218

Chapter 16. Tcl/Tk 219

Chapter 17. Networking via Ethernet 221
17.1 Ethernet Configurations 221
Standard Ethernet and Thinnet 221
Twisted-Pair Ethernet 224
17.2 Basic Ethernet Network Access 225
Ethernet Hardware Address 225
Numeric IP Address 225
Netmasks, /etc/netmasks 226
/etc/hosts File 227
/etc/hostname.*, /etc/nodename Files 228
17.3 Network Access and Security 228
Global Access Control, /etc/hosts.equiv 228
Individual Access Control, ~/.rhosts 229
17.4 Checking the Network 230
17.5 Disabling the Ethernet 231
17.6 Remote Login, Remote Shells, Remote Copy 232
17.7 Remote Printing and Plotting 232
17.8 Advanced Networking: Connecting to the Internet 233
Defining the Default Gateway (/etc/defaultrouter) 233
Activating DNS (/etc/resolv.conf, /etc/nsswitch.conf) 233
17.9 Mounting File Systems (mount, /etc/dfs/dfstab) 234
Local Mounting 234
Unmounting a Local File System 235
Remote (NFS) Mounting 235
Unmounting a Remote File System 236
NFS Mounting Options, Sharing a File System 236
Limitations 237
NFS Mounting and `root` Access 237
17.10 Automatic Mounting (/etc/vfstab) 238
17.11 Automounter 240
How it Works: An Example 240
Automount Maps (/etc/auto_master) 241
17.12 Using FTP 241
Using FTP Interactively 242
Making FTP More Automatic 243
Calling ftp Within Shell Scripts 244
17.13 NIS and NIS+ 245
17.14 Heterogeneous Networks 245
17.15 Remote Home Directories 246
Using the Automounter 246
Using Standard NFS Mounting 246
17.16 Gateway Machines 247
Spectrometer Hosts Should Not Act as Gateways! 247
17.17 Diskless Systems 248

Chapter 18. X Window System Operation 249
18.1 X Window System Environment 249

18.2 X Window System Operation 250
Starting and Exiting VnmrI 250
Starting and Exiting VnmrSGI 251
18.3 Using a Workstation as an X Server 252
Setting Up a System 252
Using the RS/6000 as an X Server 253
Using the RS/6000 as an X Server with a Sun as Host 253
Using the Sun ELC as an X Server to VnmrI 254
Using a Sun as an X Server to VnmrSGI 254
18.4 Using a Personal Computer as an X Server 254
Starting Up VNMR Using a Personal Computer 254
Setting Up Telnet 255
Starting the Client 256
Making the Display Resemble a Window Manager 256
Controlling xterm Window Appearance 257
Solving Networking Problems 257
18.5 Customizing the X Environment 258

Chapter 19. Other Networking Options 261
19.1 Hardware Options 261
19.2 Direct Connections 262
Transfers without Software Protocol 262
Using tip on a Direct Line 263
Using uucp on a Direct Line 263
19.3 Using a Modem 263
19.4 Using Mail on a Client Machine 264
19.5 Avoiding Disk Full Problems with Mail 264
Receiving Large Mail Files 264
Sending Large Mail files 264
How to Avoid Problems with Mail 265
19.6 Other Software 266

Chapter 20. Solaris 2.x vs. SunOS 4.1.x 267

Chapter 21. Calibration Tests and Shimming 269
21.1 Calibrating Pulse Width Manually 269
21.2 Testing Decoupler Field Strength 270
21.3 Testing Decoupler 90° Pulse Width with Polarization Transfer 271
21.4 Calibrating ^{1}H Decoupler Pulse Width with PPCAL 273
21.5 Calibrating ^{13}C (or X) Decoupler Pulse Width with PWXCAL 274
21.6 Shimming 275
Shim Interactions 275
Setting Low-Order (Routine) Shims 281
Removing Spinning Sidebands (Non-Routine) 282
Setting the High-Order Axial Shims (Non-Routine) 282
Setting High-Order Radial Shims (Non-Routine) 283
21.7 AutoTest—Automated Instrument Testing 284
AutoTest Step-by-Step Procedure 284
AutoTest Sample 285
Starting AutoTest 285

AutoTest Window 285
Saving Data and FID Files from Previous Runs 289
Creating Probe-Specific Files 289
AutoTest Directory Structure 290
AutoTest Macros 293
Standard Tests Performed by AutoTest 296
Details of AutoTest Experiments 299

Chapter 22. Magnet and Spectrometer Maintenance 317

22.1 Preventative Maintenance 317
Scheduled Maintenance 317
Maintenance Documentation 318
22.2 Handling Liquid Helium 318
Physical Properties of Helium 318
Handling Liquid Helium Safely 319
Measuring Liquid Helium 320
Transferring Liquid Helium 321
22.3 Continuing Dewar Service 323
Liquid Helium Service 323
Helium Refilling on Oxford 200- and 300-MHz Magnets 326
Determining Stinger Lengths for Storage Containers 326
22.4 Troubleshooting 327
Sample Scoring Problems 327
Sample Changer Troubleshooting 328
Other Troubleshooting Solutions 328

Annotated Bibliography 329

Glossary of UNIX, Sun, and VNMR Terms 337

Index 397

List of Figures

Figure 1. Typical Sun Workstation (Ultra 1) 23
Figure 2. Internal Structure of a Sun Ultra Workstation 25
Figure 3. Acquisition Processes on UNITY*INOVA* and MERCURY-VX 62
Figure 4. Acquisition Processes on MERCURY and *GEMINI 2000* 63
Figure 5. Acquisition Processes on UNITY*plus*, UNITY, and VXR-S 64
Figure 6. VNMR Accounting Window 77
Figure 7. Add Accounting Group Window 78
Figure 8. Add Group Window, Single Rate 78
Figure 9. Add Group Window, Multi Rate 79
Figure 10. Assign Holidays Window 80
Figure 11. Add User Window 80
Figure 12. Console Users Window 81
Figure 13. Console Users Window for a Selected User 82
Figure 14. Console Log Window 82
Figure 15. Printer Selection Window 83
Figure 16. UNIX File System Tree 125
Figure 17. UNIX File Storage 126
Figure 18. Structure of Directories and Subfiles 128
Figure 19. New Type of File Structure 129
Figure 20. Symbolic Link 129
Figure 21. Local Network with Tape Drive 154
Figure 22. Point-to-Point Thinnet Network 223
Figure 23. Performance Meter with Nonexistent Ethernet 224
Figure 24. Example of `app-defaults` X Resource Setup File 259
Figure 25. Measuring Residual Couplings in Dioxane 271
Figure 26. PPCAL Pulse Sequence 273
Figure 27. Approximate Shape of Axial Gradients 275
Figure 28. Theoretically Perfect Lineshape 276
Figure 29. Effects of Linear Shim Z1 276
Figure 30. Effects of Even Order (Parabolic) Shims Z2 and Z4 277
Figure 31. Effect of Odd Order (Non-Linear) Shims Z3 and Z5 278
Figure 32. Effects of Misadjusted Shims 279
Figure 33. Effects of Nonspin Shims 280
Figure 34. AutoTest Program Display 286
Figure 35. Precooling the Liquid Helium Transfer Tube 324
Figure 36. Routine Filling of the Magnet Dewar 325
Figure 37. Stinger Configuration Measurements on the Storage Container 327

List of Tables

Table 1. Performance of Sun Workstation Models 30
Table 2. Expansion Options, I/O Ports 31
Table 3. Hard Disk Characteristics. 32
Table 4. Tape Drive Characteristics 34
Table 5. Minimum Recommended Partition Sizes for Solaris 2.5.1 With CDE 42
Table 6. Minimum Recommended Partition Sizes with Solaris 2.6 43
Table 7. Format of a Nucleus Table 72
Table 8. Shell Tools and Their Properties 103
Table 9. Physical Disk Size Compared to Usable Disk Space 109
Table 10. Lock Types 116
Table 11. Usable Tape Lengths of Tape Cartridges 177
Table 12. Dump Scheme for Frequently Used Disk Partitions 177
Table 13. Examples of File Compression. 180
Table 14. Lineshape Effects and Their Associated Shims 281
Table 15. AutoTest Directories. 290
Table 16. Selected Parameters Created by ATglobal 294
Table 17. Source Code for `AT16` Macro Example 297
Table 18. Physical Properties of Helium and Nitrogen 318
Table 19. Performance Specifications for Sun CPUs 378
Table 20. Sun-3 Computer Models 382

SAFETY PRECAUTIONS

The following warning and caution notices illustrate the style used in Varian manuals for safety precaution notices and explain when each type is used:

⚠ **This symbol might be used on warning labels attached to the equipment. When you see this symbol, refer to the relevant manual for the information referred to by the warning label.**

WARNING: ***Warnings*** **are used when failure to observe instructions or precautions could result in injury or death to humans or animals, or significant property damage.**

CAUTION: ***Cautions*** **are used when failure to observe instructions could result in serious damage to equipment or loss of data.**

Warning Notices

Observe the following precautions during installation, operation, maintenance, and repair of the instrument. Failure to comply with these warnings, or with specific warnings elsewhere in Varian manuals, violates safety standards of design, manufacture, and intended use of the instrument. Varian assumes no liability for customer failure to comply with these precautions.

WARNING: **Persons with implanted or attached medical devices such as pacemakers and prosthetic parts must remain outside the 5-gauss perimeter from the centerline of the magnet.**

The superconducting magnet system generates strong magnetic fields that can affect operation of some cardiac pacemakers or harm implanted or attached devices such as prosthetic parts and metal blood vessel clips and clamps.

Pacemaker wearers should consult the user manual provided by the pacemaker manufacturer or contact the pacemaker manufacturer to determine the effect on a specific pacemaker. Pacemaker wearers should also always notify their physician and discuss the health risks of being in proximity to magnetic fields. Wearers of metal prosthetics and implants should contact their physician to determine if a danger exists.

Refer to the manuals supplied with the magnet for the size of a typical 5-gauss stray field. This gauss level should be checked after the magnet is installed.

WARNING: **Keep metal objects outside the 10-gauss perimeter from the centerline of the magnet.**

The strong magnetic field surrounding the magnet attracts objects containing steel, iron, or other ferromagnetic materials, which includes most ordinary tools, electronic equipment, compressed gas cylinders, steel chairs, and steel carts. Unless restrained, such objects can suddenly fly towards the magnet, causing possible personal injury and extensive damage to the probe, dewar, and superconducting solenoid. The greater the mass of the object, the more the magnet attracts the object.

Only non ferromagnetic materials—plastics, aluminum, wood, nonmagnetic stainless steel, etc.—should be used in the area around the magnet. If an object is stuck to the magnet surface and cannot easily be removed by hand, contact Varian service for assistance.

Refer to the manuals supplied with the magnet for the size of a typical 10-gauss stray field. This gauss level should be checked after the magnet is installed.

WARNING: **Only qualified maintenance personnel shall remove equipment covers or make internal adjustments.**

Dangerous high voltages that can kill or injure exist inside the instrument. Before working inside a cabinet, turn off the main system power switch located on the back of the console.

WARNING: **Do not substitute parts or modify the instrument.**

Any unauthorized modification could injure personnel or damage equipment and potentially terminate the warranty agreements and/or service contract. Written authorization approved by a Varian, Inc. product manager is required to implement any changes to the hardware of a Varian NMR spectrometer. Maintain safety features by referring system service to a Varian service office.

WARNING: **Do not operate in the presence of flammable gases or fumes.**

Operation with flammable gases or fumes present creates the risk of injury or death from toxic fumes, explosion, or fire.

WARNING: **Leave area immediately in the event of a magnet quench.**

If the magnet should quench (sudden appearance of gasses from the top of the dewar), leave the area immediately. Sudden release of helium or nitrogen gases can rapidly displace oxygen in an enclosed space creating a possibility of asphyxiation. Helium will displace air from the top of a room and cold nitrogen can displace air from the lower levels of a room. Do not return until the oxygen level returns to normal.

WARNING: **Avoid helium or nitrogen contact with any part of the body.**

Cold gasses or liquids (helium and nitrogen) contacting the body can cause an injury similar to a burn. Never place your head over the helium and nitrogen exit tubes on top of the magnet. If cold gasses or liquids contact the body, seek immediate medical attention, especially if the skin is blistered or the eyes are affected.

WARNING: **Do not look down the upper barrel.**

Unless the probe is removed from the magnet, never look down the upper barrel. You could be injured by the sample tube as it ejects pneumatically from the probe.

WARNING: **Do not exceed the boiling or freezing point of a sample during variable temperature experiments.**

A sample tube subjected to a change in temperature can build up excessive pressure, which can break the sample tube glass and cause injury by flying glass and toxic materials. To avoid this hazard, establish the freezing and boiling point of a sample before doing a variable temperature experiment.

WARNING: **Support the magnet and prevent it from tipping over.**

The magnet dewar has a high center of gravity and could tip over in an earthquake or after being struck by a large object, injuring personnel and causing sudden, dangerous release of nitrogen and helium gasses from the dewar. Therefore, the magnet must be supported by at least one of two methods: with ropes suspended from the ceiling or with the antivibration legs bolted to the floor. Refer to the *Installation Planning Manual* for details.

WARNING: **Do not remove the relief valves on the vent tubes.**

The relief valves prevent air from entering the nitrogen and helium vent tubes. Air that enters the magnet contains moisture that can freeze, causing blockage of the vent tubes and possibly extensive damage to the magnet. It could also cause a sudden dangerous release of nitrogen and helium gases from the dewar. Except when transferring nitrogen or helium, be certain that the relief valves are secured on the vent tubes.

WARNING: **On magnets with removable quench tubes, keep the tubes in place except during helium servicing.**

On Varian 200- and 300-MHz 54-mm magnets only, the dewar includes removable helium vent tubes. If the magnet dewar should quench (sudden appearance of gases from the top of the dewar) and the vent tubes are not in place, the helium gas would be partially vented sideways, possibly injuring the skin and eyes of personnel beside the magnet. During helium servicing, when the tubes must be removed, follow carefully the instructions and safety precautions given in the manual supplied with the magnet.

Caution Notices

Observe the following precautions during installation, operation, maintenance, and repair of the instrument. Failure to comply with these cautions, or with specific cautions elsewhere in Varian manuals, violates safety standards of design, manufacture, and intended use of the instrument. Varian assumes no liability for customer failure to comply with these precautions.

CAUTION: **Keep magnetic media, ATM and credit cards, and watches outside the 5-gauss perimeter from the centerline of the magnet.**

The strong magnetic field surrounding a superconducting magnet can erase magnetic media such as floppy disks and tapes. The field can also damage the strip of magnetic media found on credit cards, automatic teller machine (ATM) cards, and similar plastic cards. Many wrist and pocket watches are also susceptible to damage from intense magnetism.

Refer to the manuals supplied with the magnet for the size of a typical 5-gauss stray field. This gauss level should be checked after the magnet is installed.

CAUTION: **Keep the PCs, (including the LC STAR workstation) beyond the 5-gauss perimeter of the magnet.**

Avoid equipment damage or data loss by keeping PCs (including the LC workstation PC) well away from the magnet. Generally, keep the PC beyond the 5-gauss perimeter of the magnet. Refer to the *Installation Planning Guide* for magnet field plots.

CAUTION: **Check helium and nitrogen gas flow meters daily.**

Record the readings to establish the operating level. The readings will vary somewhat because of changes in barometric pressure from weather fronts. If the readings for either gas should change abruptly, contact qualified maintenance personnel. Failure to correct the cause of abnormal readings could result in extensive equipment damage.

CAUTION: **Never operate solids high-power amplifiers with liquids probes.**

On systems with solids high-power amplifiers, never operate the amplifiers with a liquids probe. The high power available from these amplifiers will destroy liquids probes. Use the appropriate high-power probe with the high-power amplifier.

CAUTION: **Take electrostatic discharge (ESD) precautions to avoid damage to sensitive electronic components.**

Wear grounded antistatic wristband or equivalent before touching any parts inside the doors and covers of the spectrometer system. Also, take ESD precautions when working near the exposed cable connectors on the back of the console.

Radio-Frequency Emission Regulations

The covers on the instrument form a barrier to radio-frequency (rf) energy. Removing any of the covers or modifying the instrument may lead to increased susceptibility to rf interference within the instrument and may increase the rf energy transmitted by the instrument in violation of regulations covering rf emissions. It is the operator’s responsibility to maintain the instrument in a condition that does not violate rf emission requirements.

Disclaimer

The information in this manual is intended to assist users in topics beyond the normal NMR spectrometer system hardware and software support provided by Varian. All information is provided on an as-is basis, and Varian support and service personnel are unable to answer questions or solve problems related to the information in this manual. For further assistance, consult local experts or textbooks in the field.

Foreword

The material for this manual was collected and written for the Varian user courses for NMR spectrometers using Sun graphics workstations as host computers. As this material has grown, it has become an important source of information for anybody dealing with such instruments: scientists, operators, Varian service personnel, and also applications scientists, including the authors.

Notice that the present version of this manual specifically refers to Solaris 2.5 (and VNMR 5.x, and 6.1, as much as VNMR specifics are involved). It is at least partially inconsistent with earlier Sun operating system and VNMR releases, in particular with SunOS 4.x. Printing has changed again with Solaris 2.6—the current version of the manual only covers Solaris 2.5 (SVR4, `lp`) printing.

This manual covers Sun Workstations and their operating system and does not cover any other computers that VNMR supports (such as the IBM RS/6000 series of graphics workstations, or SGI workstations) and their operating systems.

It must be understood that such material can never be complete nor really up to date. We are in a constant process of improving and completing it. It is hard enough to keep up with the development at VNMR, let alone Sun. Suggestions for improvement are always welcome. Chapters 16 and 20 are not written yet—they will be added in a future version of the manual. The same is true for some sections in the chapters 4, 5, and 19.

Rolf Kyburz (rolf.kyburz@ch.varian.com) &
Gerald Simon (gerald.simon@de.varian.com)
Zug and Darmstadt
February 1998

Acknowledgments

The authors would like to thank his colleagues for reviewing and verifying parts of the manual, Günter Hirtz (Varian, Darmstadt) for sharing with us his in-depth expertise in Solaris printing, and Mr. Mark Vine (Rhône-Poulenc Rorer Ltd., Dagenham, U.K.) for major contributions to the literature list as well as other chapters. Suggestions from colleagues and Varian customers that were communicated in *Magnetic Moments* or in internal publications have been adapted and incorporated into this material.

Conventions in This Manual

The following notational conventions are used throughout VNMR manuals:

- Typewriter-like characters identify VNMR and UNIX commands, parameters, directories, and file names in the text of the manual:

  The `shutdown` command is in the `/etc` directory.

- Typewriter-like characters also show text displayed on the screen, including the text echoed on the screen as you enter commands:

  ```
  Self Test Completed Successfully.
  ```

- User input is shown in usually shown in bold type:

  ```
  # cd /cdrom
  # ls
  cdrom0    solaris_2_5_1_desktop_1_1
  #
  ```

- Input or output that depends on local use is shown in italics:

  ```
  Login: root
  Password: root_password
  ```

- Optional input is shown by angled brackets:

  `seqgen s2pul<.c>` means that `seqgen s2pul.c` and `seqgen s2pul` are functionally the same.

- Lines of text containing command syntax, examples of statements, source code, and similar material are often too long to fit the width of the page. To show that a line of text had to be broken to fit into the manual, the line is cut at a convenient point (such as at a comma near the right edge of the column), a backslash (\) is inserted at the cut, and the line is continued as the next line of text. This notation is familiar to C programmers. Note that the backslash is not part of the line and, except for C source code, should not be typed when entering the line.

- Because pressing the Return key is required at the end of almost every command or line of text you type on the keyboard, use of the Return key is usually mentioned only in cases where it is *not* used. This convention avoids repeating the instruction “press the Return key” throughout most of this manual.

Chapter 1. Sun Hardware

Sections in this chapter:

- 1.1 “How Does a Workstation Work?,” this page
- 1.2 “Sun Workstations—An Overview,” page 28
- 1.3 “Data Storage Options,” page 29
- 1.4 “Installing the Hardware,” page 35
- 1.5 “Adding a Disk (Hardware),” page 35

Welcome to the world of Sun, Solaris and VNMR! In this chapter, we start our study of system administration by discussing Sun workstations and hardware installation:

1.1 How Does a Workstation Work?

A Sun workstation (Figure 1) is a powerful desktop computer:

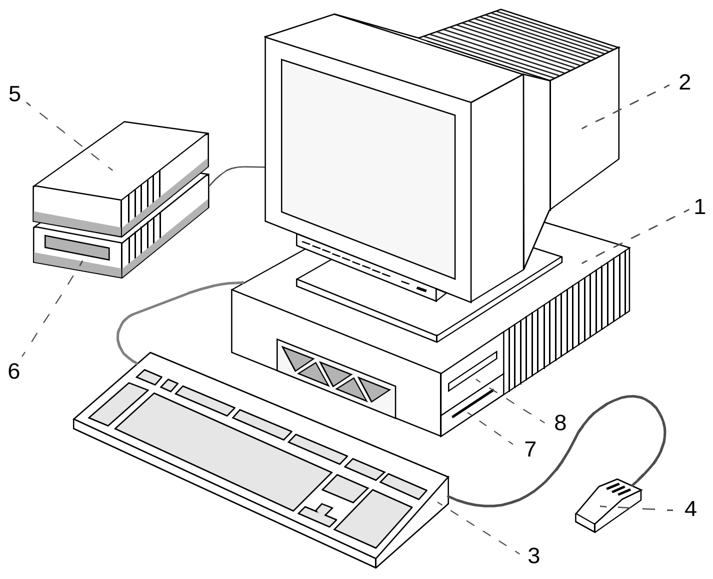


Figure 1. Typical Sun Workstation (Ultra 1)

You see a few units standing on your desk: the computer itself (the CPU module, 1), the CRT screen (2), the keyboard (3), and the mouse (4). Some systems are equipped with external storage modules, Sun now calls them unipacks, containing things such as an external disk drive (5) or an external tape drive (6). External in this context means outside the CPU module.

CPU Module

The CPU module itself contains most of the computer. The outside only reveals the peripherals connectors in the back, the loudspeaker opening on the front side, plus on some systems the access to an internal floppy drive (7) and an internal CD-ROM (8). Some systems also have the CD-ROM drive in an external module.

The CPU module is electrically shielded and contains several subunits, such as

- The actual CPU module, a single printed circuit board with the CPU chip, memory, plus I/O controllers, bus and I/O connectors
- Graphics controller
- Internal hard disks
- Optional expansion modules
- Power supply and cooling fans

plus the parts that are visible or accessible from outside, such as

- Floppy disk drive (optional)
- CD-ROM drive (optional)
- Loudspeaker

However, the external structure of a workstation does not reveal much about its internal functionality. Figure 2 is a schematic view of the internal structure of a modern Sun workstation (we have taken the Sun Ultra series as the model):

Most computer functions are built into a single printed-circuit board (PCB), the CPU board. In Figure 2, this includes the CPU (on some systems the CPU is on a pluggable, separate module), the memory, the system interconnect, the random-access memory (RAM), and all the controller functions, in most cases excluding the graphics controller.

Let’s discuss the system functionality in a bit more detail.

CPU Chip

The CPU chip combines many functions that in earlier systems were spread over several separate components (chips). Modern Sun computers are equipped with one of the following CPU chips:

- MicroSPARC or TurboSPARC (SPARCstation 4, SPARCstation 5)
- SuperSPARC (SPARCstation 10, SPARCstation 20)
- HyperSPARC (some SPARCstation 20 models)
- UltraSPARC (Ultra workstations)

These CPU chips include floating point calculation facilities (FPU), virtual memory management (MMU), and very fast primary cache memory. The UltraSPARC chip even includes some dedicated graphics functions. Earlier models has separate floating point coprocessor (FPC) and MMU chips. The primary cache memory is integrated into the CPU, making it very fast. The CPU tries to avoid access to the slower main memory or the disk by keeping the most current data and instructions in the primary cache.

Some models (SPARCstation 10/41, 10/52, SPARCstation 20; all Ultra systems) are also equipped with between 256 KB and 2 MB of secondary (level 2) cache memory, usually on a separate chip. A few models (SPARCstation 10, SPARCstation 20, Ultra 2, Ultra 60, high-end servers) can also be equipped with multiple CPU chips (on plug-in CPU modules). The Ultra IIi CPU (used in the Ultra 5, Ultra 10) also includes a PCI bus controller.

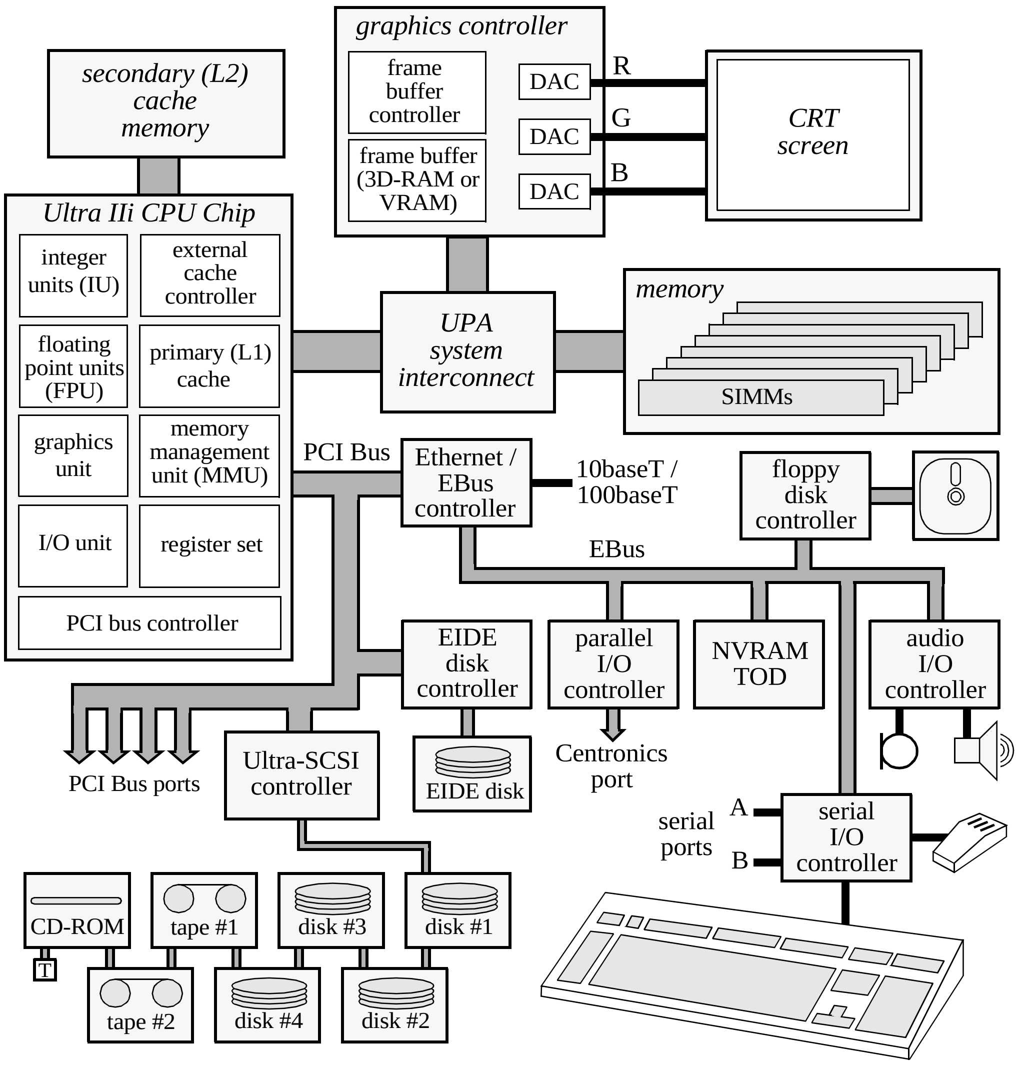


Figure 2. Internal Structure of a Sun Ultra Workstation

Random-Access Memory

The random-access memory (RAM) is implemented in the form of small pluggable memory modules. Each has 1 MB to 64 MB of memory with parity checking, or even ECC (error checking and correction) memory on high-end models. All current models are equipped with 64 or more MB of RAM—earlier models had only 8, 16, 24, or 32 MB. When running Solaris 2.x and VnmrX, systems with less than 32 MB of RAM show noticeably slower reaction times, because they use the disk much more frequently.

Graphics Controller

On SPARCstations, the graphics controller is connected via a SBus or PCI bus port (see below). In high-end Sun Ultra workstations, the graphics controller (Creator, Creator3D or Elite3D) is integrated via the main system interconnect structure (the UPA, see below), which allows for faster data transfer rates and therefore much faster graphics.

Several years ago, Sun sold low-end SPARCstation models with a black and white screen (SPARCstation SLC and ELC). In the meantime, all models are equipped with a color screen. Therefore, the graphics controller has a frame buffer controller (local processor for manipulating the display) and a frame buffer (memory with the display information) with 8 or more bits per pixel for storing the color information. 8 bits per pixel allow for 256 different colors. The colors are defined by numeric (8-bit) amplitudes for each of the color components (red, green, blue), allowing for $256 \times 256 \times 256 = 16{,}777{,}216$ different colors. On 8-bit color systems, a lookup table associates each of the 256 different color numbers from the frame buffer with one of the 16,777,216 possible real colors.

High-end graphics systems, such as the SX graphics accelerator in some SPARCstation 20 systems and the Creator, Creator3D, and Elite3D graphics controllers for the high-end Ultra, use 24 and more bits per pixel and don't need a lookup table. Sun screens have 900×1152, 1024×1280 or more pixels; therefore, an 8-bit frame buffer is 1 MB or 1.25 MB. High-end graphics systems use much larger frame buffers.

System Interconnect

A central point for the overall system performance is the system interconnect, because internal communication may well be a bottleneck for certain system performance aspects. The high-end Ultra workstations use a structure called UPA (UltraSPARC Port Architecture) that allows for fast (1.3 GB/sec) multipath communication between the different system components. SPARCstation 10 and SPARCstation 20 systems use the MBus for connecting CPU modules, instead of UPA.

PCI Bus / SBus

On the current Ultra workstations, the system I/O is done via PCI bus, which serves as a primary backbone for connecting all I/O devices. On systems with Ultra IIi CPU, the PCI bus is controlled by the CPU chip itself; on others (Ultra 30, Ultra 60), it uses a separate controller attached to the UPA system interconnect. On SPARCstations and early Ultra workstations (Ultra 1, Ultra 2) the Sun SBus was used instead of the PCI bus (which is less expensive to build). These facilities are built into the CPU board:

- *SCSI / UltraSCSI controller* – Most systems include some internal SCSI devices, such as one or two internal disks, an optional internal CD-ROM or tape drive, as well as an external SCSI connector for connecting external storage devices, such as tape drives, extra disks, a CD-ROM, or other optical disks. On SPARCstations and the low-end Ultras, the SCSI bus operates at up to 10 MB/sec and allows for up to 7 devices. The software is preconfigured for four disks, up to two tape drives and one CD-ROM.

  Ultra 1 systems with Creator graphics (such as the Ultra 1/170E) are equipped with a fast/wide SCSI interface (SCSI-3, 16-bit bus width, as opposed to an 8-bit width on the standard SCSI interface), which allows up to 15 devices operating at transfer rates of up to 20 MB/sec. Newer Ultra systems (Ultra 30, Ultra 60) are equipped with a 40 MB/sec UltraSCSI port.
- *Ethernet / EBus controller* – All systems are equipped with an AUI Ethernet transceiver port. Some older systems also have a BNC thin Ethernet connector (transceiver built into the CPU board). Newer systems are equipped with a twisted pair (TP) port (10BaseT). The Ethernet interface operates at 10 Mbps.

  Ultra systems with Creator graphics (such as the Ultra 1/170E) are equipped with a 100BaseT twisted pair "fast Ethernet" port operating at 100 Mbps. On newer Ultra systems, all the lower speed I/O is done via EBus, which is controlled by the same hardware as the Ethernet port.

- *PCI ports* – PCI ports (SBus ports on systems with SBus) are used for optional peripherals (such as low-end and third party graphics controllers), a second SCSI interface (also fast/wide SCSI is available), a second Ethernet interface, extra serial and parallel I/O ports, and ISDN ports. Even PC coprocessors are available on SBus or PCI cards. Most systems come with several such expansion ports.
- *ISDN port* – An ISDN port is included on SPARCstation 10 systems only, for data connections over digital telephone networks.

EBus

On current Ultra workstations, lower- and medium-speed I/O is done via dedicated EBus (external bus) interface, which connects several facilities built into the CPU board. On SPARCstations, these facilities are connected to the SBus as well.

- *EIDE disk controller* – Lower end Ultra workstations (Ultra 5, Ultra 10) are equipped with 1 or 2 EIDE disks.
- *Floppy disk controller* – Needed for an optional internal floppy disk drive (1.44 MB).
- *Parallel I/O controller with Centronics port* – Typically used for printers/plotters.
- *Serial I/O controller* – Used with two RS-423 (RS-232 compatible) serial ports for external peripherals, such as printers and plotters, plus two internal ports used for the keyboard and the mouse.
- *Audio I/O controller* – For driving the built-in loudspeaker (optional on some systems) and taking input from a microphone port.
- *NVRAM, TOD* – On Ultra workstations, nonvolatile RAM (NVRAM; EEPROMS on earlier workstations) and the real-time clock (TOD) are accessed via the EBus.

Other CPU Board Components

The CPU board contains additional components less visible to the user, such as PROMs containing diagnostics software and system bootup utilities, an EEPROM (nonvolatile memory, NVRAM) for storing system configuration parameters, and quartz oscillators that generate the operating frequencies for some of the components. Unless the CPU box is open, the user can usually only access the connectors for the peripherals on the front of the CPU board.

On some systems, there is not enough space for all the connections along the front of the CPU board. In these cases, some connectors combine two ports, in which case one of the two functions is only available via a special “V-shaped” branch cable. Some SPARCstation models have two serial ports combined in one V.24 connector, and some combine the AUI Ethernet and audio ports in a single connector.

SCSI Bus

The SCSI bus is a linear chain that has a maximum total length of about 5 meters by its electrical definition, including all internal cabling. The SCSI bus must be terminated at the remote end to work properly. With desktop storage modules, a terminator plug must be plugged into the second connector of the last module in the chain. The newer Unipack modules are self-terminating—a termination plug is no longer necessary. SCSI devices can be connected in any order (with certain very specific exceptions as discussed below).

The current desktop SCSI expansion modules, the Unipack, as well as some Ultra workstations are equipped with SCSI-3 (fast/wide SCSI-2) connectors, even if the devices involved are SCSI-2 only. This raises some cabling issues:

- To connect Unipack expansion modules with any SPARCstation, a 50- to 68-pin conversion cable is required.
- To connect older desktop storage modules with SCSI-2 connector, a 50- to 68-pin conversion cable is required.
- Most Ultra workstations are equipped with 68-pin SCSI-3 or UltraSCSI connector; here, only 68- to 68-pin cables should be required.
- High-end Ultra workstations are equipped with an internal fast (7200 rpm) SCSI-3 or UltraSCSI disk. If SCSI-2 and SCSI-3 devices are to be connected to such a system, the slower SCSI-2 devices must be at the end of the SCSI chain. All Sun tape drives and CD-ROM drives are SCSI-2 only.

Each device on the SCSI bus requires a unique bus address ranging from 0 to 7 (SCSI-2) or 0 to 15 (SCSI-3, UltraSCSI).

- On UNITY*plus* spectrometers, address 2 is reserved for acquisition.
- On UNITY and VXR-S spectrometers address 3 is reserved for acquisition[1].

The addresses 0 to 3 (with the exception of the acquisition addresses on the spectrometers mentioned above) are typically used for disks, addresses 4 and 5 for tape drives, address 6 for a CD-ROM (where available), and address 7 for the SCSI controller.

The Ultra 1/170 and other systems with SCSI-3 or UltraSCSI controller allow for 8 additional devices (addresses 8 to 15). If more devices are to be connected, an additional SCSI controller (SBus or PCI bus card) is required[2].

1.2 Sun Workstations—An Overview

Many different Sun workstations have been—and still are—used in Varian NMR spectrometers and as NMR processing workstations. The tables in this section should give you an idea where in the “evolution history” your own workstation is situated.

Note that the Sun-3 models are no longer supported by Solaris 2.x, the current Sun operating system, and some early Sun-4 models have only limited support (floating point hardware not supported under Solaris 2.x, which makes them unusable for NMR processing). These systems are only listed here for comparison purposes. In fact, these rather lengthy tables cover only ten years of workstation history.

[1] Early UNITY and VXR-S systems were equipped with a HAL board without DMA (direct memory access) capability. If you are upgrading the host from an early workstation model, note that the SPARCstation 2, the SPARCstation IPX, and any Sun workstation after these models *require* a HAL board with DMA capability. This includes some modifications to the SCSI differential driver box.

[2] On UNITY*plus*, UNITY, and VXR-S spectrometers, the acquisition computer is connected to the Sun via the SCSI port. In order to be able to use a cable of reasonable length between the acquisition cabinet and the Sun, Varian uses a SCSI driver box that extends the SCSI bus over much longer distances. Electrically, the acquisition extension is not the same bus as the Sun SCSI bus. This means that if the driver box is located at the end of the SCSI chain (excluding the acquisition), it must still be terminated internally, even with the acquisition CPU connected. From a software point of view, the acquisition CPU (the Host-to-Acquisition, or HAL, more exactly) is not a standard SCSI device. Special software (a so-called device driver) is used to communicate with the console. In UNITY*INOVA*, *MERCURY*-Series, and *GEMINI 2000* spectrometers, the communication with the acquisition is established via Ethernet.

CPU Type, Performance

The CPU types used in Sun workstations range from the Motorola 68000 family (68020, 68030) to the family of SPARC microprocessors (including the SPARC, MicroSPARC, TurboSPARC, SuperSPARC, HyperSPARC, and UltraSPARC). Some early systems (Motorola- and SPARC-based) required floating point coprocessors.

Over the past ten years, the processing power of these systems has been increased by a factor of hundred and more (see Table 1). It is impossible to compare these computers using a single performance measure. Early benchmark tests (such as MIPS and MFlops) run unrealistically fast on modern computers, because the test programs are small enough to fit into a computer's fast cache memory. On the other hand, early computers are not powerful enough (in terms of CPU speed, memory size, etc.) to run the modern benchmark tests (such as the SPECint and SPECfp). For more information on these benchmark tests, see "Glossary of UNIX, Sun, and VNMR Terms," page 337.

Expansion Options, I/O Ports.

The workstations listed in Table 1 not only differ in their processor model and speed—they also differ in the number and kind of I/O ports, and in their expansion capabilities, and those often also play a role in the decision which workstation model to buy (or which model to upgrade to). Table 2 lists only workstations that are fully supported by Solaris 2.x.

Note that on networked UNITY*INOVA* and *GEMINI 2000* spectrometers, one additional SBus slot is taken by the second Ethernet interface board. On SPARCstation 4 systems, that second Ethernet interface board fills up the only SBus slot.

1.3 Data Storage Options

It seems to be an unwritten law in computing that no matter how much disk space you purchase with the system, you always run out sooner than you think possible. At that point most users need to look into expanding the disk space on their workstation. Often, there is also a need for storing data on transportable media. A wide choice of media options are available, and in this section we discuss the various data storage options.

Hard Disk Drives

Over the years, not only has the capacity of the hard disks used in Sun workstations increased dramatically, but also parameters such as the average seek time, rotational latency (i.e., rotation speed), and maximum data transfer rates have improved, while the disk drives have become smaller and smaller. Upgrading your system with a newer disk not only increases the storage capacity—it should also improve the performance.

Table 3, lists some of the characteristics of disk drives used in SPARCstation and Ultra workstations. For an explanation of the terms used in this table, see the section "Glossary of UNIX, Sun, and VNMR Terms," page 337.

One internal disk drive is standard with all Sun workstations currently sold through Varian. SPARCstation and Ultra computers are sold with a 7200 rpm 2.1-GB internal hard disk, but only Ultra systems with Creator graphics (1/170E, 1/200E) are equipped with a SCSI-3 interface.

Table 1. Performance of Sun Workstation Models

Workstation model	*CPU / FPC*	*CPU MHz*	*Cache MB*	*Integer performance*			*Floating point performance*		
				MIPS	*SPECint92*	*SPECint95*	*MFlops*	*SPECfp92*	*SPECfp95*
Sun 3/50	68020 / 68881	15	0	1.5			0.09		
Sun 3/110, 3/150, 3/160	68020 / 68881	17	0	2.0			0.11		
Sun 3/110, 3/150, 3/160	68020 / FPA	17	0	2.0			0.40		
Sun 3/60	68020 / 68881	20	0	3.0			0.13		
Sun 4/110	SPARC / Weitek	14	0	7.0			0.80		
Sun 4/260	SPARC / Weitek	17	0	10.0			1.1		
SPARCstation SLC	SPARC / Weitek	20	0	12.5	9.9		1.2	8.1	
SPARCstation ELC	SPARC / Weitek	33	0	21.0	18.2		3.0	17.9	
SPARCstation IPC	SPARC / Weitek	25	0	17.4	13.8		1.7	11.1	
SPARCstation IPX	SPARC	40	0	28.5	21.8		4.2	21.5	
SPARCstation 1	SPARC / Weitek	20	0	12.5			1.4		
SPARCstation 1+	SPARC / Weitek	25	0	15.8			1.7		
SPARCstation 330	SPARC / TI FPU-2	25	0	16.0			2.5		
SPARCstation 2	SPARC	40	0	28.5	21.8		4.2	22.8	
SPARCclassic	SPARC	50	0	59.0	26.4		4.6	21.0	
SPARCstation LX	SPARC	50	0	59.0	26.4		4.6	21.0	
SPARCstation 4/70, 5/70	MicroSPARC	70	0	100	57.0		13.1	47.3	
SPARCstation 4/85, 5/85	MicroSPARC	85	0	112	64.0		14.9	54.6	
SPARCstation 4/110, 5/110	MicroSPARC	110	0	135	78.6	1.59	21.7	65.3	1.99
SPARCstation 5/170	TurboSPARC	170	0.5			3.32			2.91
SPARCstation 10/30	SuperSPARC	36	0	102	45.2		20.5	54.0	
SPARCstation 10/40	SuperSPARC	40	0	110	50.6	1.13	22.9	60.2	1.38
SPARCstation 10/41	SuperSPARC	40	1	110	53.2		22.4	67.8	
SPARCstation 10/51	SuperSPARC	50	1	135	65.2		27.3	83.0	
SPARCstation 20/50	SuperSPARC	50	0	134	76.9		30.5	80.1	
SPARCstation 20/51	SuperSPARC	50	1	133	81.8		30.5	89.0	
SPARCstation 20/61	SuperSPARC	60	1	167	98.2		36.6	107	
SPARCstation 20/71	SuperSPARC II	75	1	205	126	3.11	44.4	121	3.10
SPARCstation 20/151	HyperSPARC	150	0.5		169			208	
Ultra 1/140	UltraSPARC	143	0.5	291	215	5.41	109	303	7.90
Ultra 1/140E	UltraSPARC	143	0.5			5.87			8.38
Ultra 1/170	UltraSPARC	167	0.5	341	252	5.56	126	351	9.06
Ultra 1/170E	UltraSPARC	167	0.5	341	252	6.26	126	351	9.06
Ultra 1/200E	UltraSPARC	200	0.5			7.44			10.4
Ultra 5	UltraSPARC IIi	270	0.25			9.1			10.1
Ultra 10	UltraSPARC IIi	300	0.5			12.1			12.9
Ultra 30/250	UltraSPARC II	250	1			10.0			14.9
Ultra 30/300	UltraSPARC II	300	2			12.1			18.3
Ultra 60/1300	UltraSPARC II	300	2			13.0			18.3
Ultra 60/2300 (2 CPUs)	UltraSPARC II	300	2			13.1			23.5
Ultra 60/1360	UltraSPARC II	360	4			16.1			23.5
Ultra 60/2360 (2 CPUs)	UltraSPARC II	360	4			16.1			29.5

Table 2. Expansion Options, I/O Ports

Workstation	*max. RAM MB*	*Graphics*[a]	*Floppy*[b]	*int. CD-ROM*[c]	*int. Harddisks*[d]	*Expansions*				*I/O ports*			*Ethernet*		
						SBus[e]	*MBus*[f]	*PCI*[g]	*UPA*	*serial*	*parallel*	*ISDN*	*AUI / MII*	*10BaseT*	*100BaseT*[h]
SPARCstation SLC	16	b&w								2			1		
SPARCstation ELC	64	b&w								2			1		
SPARCstation IPC	48	8-bit	1		1	2				2			1		
SPARCstation IPX	64	GX	1		1	2				2			1		
SPARCstation 1/1+	64	GX	1		1	2				2			1		
SPARCstation 2	128	GX	1		1	2				2			1		
SPARCclassic	96	8-bit	1		1	2				2	1		1	1	
SPARCstation LX	96	GX	1		1	2				2	1	1	1	1	
SPARCstation 4	160	8-bit	1	1	1	1				2	1		1	1	
SPARCstation 5	256	TGX	1	1	2	2				2	1		1	1	
SPARCstation 10	512	GX	1	1	2	3	2			2	1	1	1	1	
SPARCstation 20	1024	TGX	1	1	2	3	2			2	1		1	1	
SPARCstation 20	1024	SX[i]	1	1	2	4	2			2	1		1	1	
Ultra 1	1024	TGX	1	1	2	3				2	1		1	1	
Ultra 1	1024	Creator[j]	1	1	2	2			1	2	1		1		1
Ultra 5	512	PGX	1	1	1			3		2	1				1
Ultra 10	1024	various	1	1	2			4	1	2	1				1
Ultra 30	2048	Creator	1	1	2			4	2	2	1		1		1
Ultra 60	2048	various	1	1	2			4	2	2	1		1		1

a. The graphics hardware listed is the basic option sold through Varian; most models can be upgraded with higher performance graphics. All options listed are 8-bit color graphics controllers, with the exception of b&w (black & white graphics), SX (24-bit), and Creator (24-bit).

b. The floppy drive is always optional.

c. The internal CD-ROM drive is always optional.

d. One internal disk is standard, Ultra 1 systems are equipped with fast/wide SCSI disks, Ultra 5 and 10 systems are equipped with EIDE disks, Ultra 30 and 60 systems are equipped with UltraSCSI disks.

e. Number of *free* Sbus slots with graphics option listed (1 slot may be taken by graphics controller).

f. Total number of MBus slots (for exchangeable / upgradable CPU modules). Models with MBus option can be upgraded to a faster CPU module, or to a multiprocessor system without changing the motherboard.

g. Full or half size PCI slots, 33 MHz or 66 MHz, depending on model.

h. 100BaseT fast Ethernet is compatible with 10BaseT, standard Ethernet.

i. We used 24-bit SX graphics in the SPARCstation 20/71 and 20/151 models.

j. 24-bit Creator graphics is standard with the Ultra 1/140E, 1/170E and 1/200E models.

Floppy Disk Drives

All desktop SPARCstations (except for the SPARCstation SLC and the SPARCstation ELC) and Ultra systems can be equipped with an optional internal 3.5-in. floppy disk drive. Floppies can be in the `tar` and UNIX format (1.44 MB), or in the DOS format (720 KB or 1.44 MB). For more information, see "Using Floppy Disks," page 156.

Table 3. Hard Disk Characteristics.

Disk type	*Form factor, in.*	*Capacity, MB*	*Average read seek time, msec*	*Rotational latency, msec*	*Rotation speed, rpm*	*Max. transfer rate, MB/sec*
Sun 327 MB SCSI	5.25	327	16.5	8.3	3600	1.5
Sun 669 MB SCSI	5.25	669	16.0	8.3	3600	1.8
Sun 1.3 GB SCSI	5.25	1300	12.0	5.5	5400	4.5
Sun 104 MB SCSI	3.5	104	22.0	8.2	3660	1.2
Sun 207 MB SCSI	3.5	207	16.0	8.3	3600	1.6
Sun 424 MB SCSI	3.5	424	14.0	6.8	4400	3.0
Sun 535 MB SCSI-2	3.5	535	12.0	5.5	5400	5.0
Sun 1.05 GB SCSI-2	3.5	1050	10.5	5.5	5400	5.9
Sun 2.1 GB SCSI-2	3.5	2100	10.0	5.5	5400	5.8
Sun 2.1 GB SCSI-3	3.5	2100	9.5	4.2	7200	8.8
Sun 4.3 GB EIDE	3.5	4300	10.0	5.5	5400	
Sun 4.2 GB SCSI-3[a]	3.5	4200	11.0	5.5	5400	7.6
Sun 4.2 GB SCSI-3	3.5	4200	9.5	4.2	7200	8.8
Sun 9.1 GB SCSI-3	3.5	9100	9.5	4.2	7200	8.8

a. In early 1997, this disk type was replaced with the faster 7200 rpm drive.

CD-ROM Drives

Solaris and VNMR software is only available on CD-ROM (644 MB read-only, 5.25-in. optical disks). For loading software, at least one CD-ROM drive (644-MB read-only storage) should be available per site.

For early SPARCstations (up to the SPARCstation 10), a CD-ROM was only available in an external storage module. These were single-speed (maximum transfer rate 1.2 MB/sec) drives that required a caddy to insert the CD-ROM. Early SPARCstation 4, 5, and 20 systems could be equipped with an internal double-speed CD-ROM drive (without caddy). This drive was discontinued by the manufacturer around mid-1996, and Sun switched to an internal quad-speed drive.

Because the form factor for these new drives is different, early SPARCstation 4, 5, and 20 systems cannot be retrofitted with a new internal CD-ROM drive (the older drives are no longer available). In these cases, an external CD-ROM drive must be selected. Newer SPARCstations and the Ultra workstations can be equipped with an internal quad-speed CD-ROM drive. An external CD-ROM drive can be added to any SPARCstation or Ultra system.

In 1997, the quad-speed Sun CD-ROM drives were replaced with 12X drives (12 times the speed of the first generation CD-ROM drives). This drive has an average transfer rate of 1.8 MB/sec (10 MB/sec burst rate) and a longer MTBF of 100,000 hours.

The latest Ultra workstations are equipped with 24X CD-ROM drives (24 times the speed of the first generation CD-ROM drives), the standard in the high-end PC world.

Tape Drives

Tape drives have also seen considerable development. Early Sun-3 and Sun-4 systems were equipped with a 60-MB quarter-inch cartridge (QIC) drive QIC-24). Later, SPARCstations were equipped with 150 MB (QIC 150) drives that could read (but not write) the previous QIC-11 (25 MB) and QIC-24 tape formats. For many years such tapes were used to load SunOS and VNMR software. For some time, VNMR was also available on 8-mm Exabyte tape (2.5 GB format).

Starting with SunOS 4.1.2, the Sun operating system is only available on CD-ROM, and VNMR 5.1B and later releases are also only available on CD-ROM. Therefore, tapes are now used exclusively for backup, data archiving, and data exchange and transport.

There was further development in the area of QIC tape drives. The standard QIC tape drive from Sun now allows for a capacity of up to 2.5 GB per tape (QIC-2000). This raises some compatibility issues because the newer tape drives so far can still read all the lower density formats, but in general it is not possible to write in lower QIC density formats—with two exceptions: 60 MB (QIC-24) drives can also write in QIC-11 format (this is for Sun-3 and early Sun-4 systems only) and 2.5 GB (QIC-2000) drives can write in 150-MB (QIC-150), 525-MB (QIC-525) and 1.2-GB (QIC-1000) formats.

There are also other tape drive options:

- *8-mm Exabyte (video technology) tape drives* – The first version of these tape drives offered a capacity of 2.5 GB. In later versions, the capacity was increased to 5 GB, then 10 GB, and finally 14 GB. All these drives can read and write lower density or capacity tape formats. The 5 GB and higher capacities are achieved through data compression hardware in the tape drive. For data that are already compressed, typically 50% of the nominal capacity is achieved (2.5, 5, and 7 GB, respectively).
- *4 -mm DDS-2 / DAT (digital audio tape) drives* – These drives, with 4-GB native tape capacity (8 GB with compression), are also available from Sun. Before that, Sun was offering a DAT tape drive with 2 GB native capacity (up to 5 GB with compression).

Because loading software via tape drive no longer is an issue, the user is now free to chose among the various tape options. This decision should be based on the following factors:

- Compatibility with existing tape archives, tape drives, and the need for data exchange with other systems via tape.
- Reliability of the drive, and the durability of the media (QIC wins here).
- Tape capacity compared to the storage needs and the size of the installed disk drives.
- Data transfer speed, particularly for large backups (8-mm Exabyte drives offer the highest transfer rates).
- Costs for the tape drive (8 mm-drives are more expensive).
- Costs for the media (QIC tapes are relatively expensive).
- Space consumption for large tape archives (DAT and Exabyte take less space).

For your convenience, Table 4 lists tape drive characteristics.

Note that we strongly recommend to use computer-grade 4-mm and 8-mm tapes, not just standard audio or video quality. It does not make sense to calculate and compare media prices in costs per MB, because in most cases only a small fraction of the tape capacity is actually used. It is far more important to have data secured on tape than to fill a tape to 100% (especially considering the low price of media).

If you are unsure about the type of 8-mm Exabyte tape drive installed on your system, the following descriptions might help:

Table 4. Tape Drive Characteristics

Model	*Capacity, native*	*Capacity, with compression*	*Transfer rate KB/sec, native*	*Transfer rate KB/sec, w/compr.*	*MTBF, hours*	*Drive costs, est. U.S. $*	*Media type*	*Media costs, U.S. $*[a]
QIC-150	150 MB	n.a.[b]	110	n.a.			DC 6150	12
QIC-2000[c]	2.5 GB	n.a.[d]	300	n.a.	200,000	1500	DC 9250	40
DAT	2 GB	5 GB	366	800	80,000		90 m	10
DDS-2 (DAT)[e]	4 GB	8 GB	400	800	80,000	1800	120 m	25
DDS-2 (DAT)	4 GB	8 GB	550	1100	80,000	1800	120 m	25
DDS-3 (DAT)[f]	12 GB	24 GB	1000	2000		3200	125 m	
Exabyte 8200	2.3 GB	n.a.[g]	250	n.a.			112 m	10
Exabyte 8500	2.5 GB	5 GB	250	500	40,000		112 m	10
Exabyte 8505	5 GB	10 GB	500	1000	160,000		112 m	10
Exabyte 8505XL[h]	7 GB	14 GB	500	1000	160,000	3500	160 m	20
Exabyte 8505XL	7 GB	14 GB	1000	2000	160,000	3500	160 m	20
Exabyte 8900[i]	20 GB	40 GB	3000	6000	200,000	8500	170 m	8

a. Media costs are estimated street prices.

b. With software compression (such as with the UNIX `compress` command), up to about 300 MB can typically be stored on a single cartridge.

c. Besides QIC-150, this tape drive can also read and write tapes in QIC-525 format (525 MB with 1020-foot tape) and QIC-1000 format (1.2 GB with 950-foot tape). With 950-foot tape (instead of 1200-foot for 2.5 GB) the capacity is 2 GB. The media type listed is for the highest capacity.

d. With software compression, up to about 5 GB can be stored on a single cartridge.

e. Replaced with the faster version around mid-1997.

f. Currently not part of the standard Varian offering, requires special tape quality.

g. With software compression up to about 5 GB can be stored on a single cartridge.

h. Replaced with the faster version around mid-1997.

i. Not part of the standard Varian offering, mainly used in servers.

- The Exabyte 8200 (2.3 GB) is a full-height 5.25-in. SCSI device in a double-height desktop enclosure with two LEDs aligned horizontally on the front panel
- The Exabyte 8500 (5 GB) is similar, but the two LEDs are aligned vertically. The “medium-density” mode enables the hardware compression and decompression.
- The 10-GB and 14-GB drives are half-height devices and fit into a standard Unipack desktop enclosure. The Exabyte 8505 (10 GB) rejects 160-m 14-GB media (10-GB drives were only sold for a few months). 10- and 14-GB capacity (including hardware compression) is addressed using the “high-density” mode. See Chapter 10, “Using Tapes and Floppy Disks,” for further information.

To avoid problems, you should use the 112-m media. These tapes are accepted by all Exabyte drives (10-GB capacity seems more than enough), and the 112-m tapes are sold at about half the price of 160-m (14-GB) tape.

Read/Write Optical Disk Drives, WORM Drives

As an alternative to tape drives, read/write optical disk drives are available from third-party vendors. Or for long-term archiving (over ten years), third-party vendors also sell WORM (write once, read many) optical disk drives (typical capacity 2×300 or more MB per disk). Some of these drives approach the speed of a slow magnetic hard disk for reading data. Typically, writing data takes twice as long as reading. Writing involves local heating under an external magnetic field, using a laser beam that is stronger than the beam used for reading the disk. WORM drives are the ideal choice for compliance with the GLP requirements for long-term, secure data archiving, because the data cannot be altered once it is on a WORM disk.

The emerging standard in writable optical media seems to be the writable CD which is now available for PCs and for UNIX workstations. This permits writing optical disks in standard CD-ROM format. So far, no such drive is available from Sun; you need to refer to third party manufacturers.

1.4 Installing the Hardware

Should you install a Sun system yourself, here are a few reminders for a safe installation:

- Current Sun workstations and their peripherals are equipped with power supplies that can be plugged into any power outlet in the range of 110 to 240 Vac and 50 to 60 Hz without any switching.
- Never insert or remove boards unless the power to the computer is first switched off, but leave the power cord plugged in to ensure proper grounding.
- If you want to install a board or an extension card, or add more memory, never touch any internal component without antistatic protection.
- If you need to carry the CRT screen, first switch it off, and then wait for a several minutes before disconnecting the power cord; otherwise, the internal capacitors may not completely discharge, and you might experience irritating power shocks when lifting the screen around!

1.5 Adding a Disk (Hardware)

The most frequently selected hardware expansion is the addition of a new hard disk drive. This is somewhat more complex than adding memory because it involves changes to the SCSI bus configuration, and it involves a number of software operations, including a partial reorganization of the disk file systems (see "Adding a Disk (Software)," page 51 for the corresponding software manipulations).

Additional disks can be installed by the user. There are two different cases: internal disks and external disks. An external disk can be added to any system, but a second internal disk can only be added to more recent systems.

Adding an Internal Disk

A second internal disk can be added to SPARCstation 5, 10, and 20, and Ultra computers:

- *SPARCstation 10* – Select a SCSI ID address by setting a jumper according to the installation instructions that come with the disk drive. Select a SCSI ID between 0 and 3 that isn't already in use. Typically, the first drive has address 3 (c0t3d0).[3]

- *SPARCstation 5 or 20, or Ultra systems* – The SCSI ID is "wired" into the SCSI connector, making the SCSI ID selection automatic and instantaneous with the insertion of the drive. The automatic address for a second internal disk drive is 1 (c0t1d0), so you must make sure there is no external drive with that address. To install the disk, follow the instructions that come with the drive. Be sure to take antistatic measures when working inside the CPU box.[4]
- *Ultra 10* – This workstation can be equipped with a second internal EIDE disk drive.
- *Ultra 5 and Ultra 10* – External SCSI drives can be added only through an optional SCSI expansion card on the PCI bus.

After installing the drive, close the CPU box, reconnect all the peripherals, and boot the system with the `boot -r` command.

Adding an External Hard Disk

Adding an external disk is much easier.

1. After switching power off to the computer, insert the drive into the external SCSI chain or connect it to the SCSI connector at the back of the CPU.
2. Select a SCSI ID between 0 and 3 that isn't used already. You may want to use the `dmesg` command for shutting down your computer—this tells you which SCSI addresses are taken already.[5]

   On Ultra systems and SPARCstation 5 and 20 computers, it is advisable not to use addresses 1 and 3 because these are reserved for internal drives. Using these addresses may make it difficult to add a second internal disk drive at a later date.
3. Turn your computer back on again and boot it with the `boot -r` command.

Checking the SCSI Address

After installing the new hard disk, you can verify the SCSI address settings from the monitor (PROM diagnostics) software.

1. It is best to interrupt the bootup process after switching the workstation back on, as soon as you get the first bootup messages on the screen, by pressing Stop-a (i.e., pressing the Stop and A keys simultaneously).
2. If you get the ">" prompt, switch to the "`ok`" prompt for the new monitor mode by typing `n`:

   ```
   > n
   ok
   ```

[3] On UNITY*plus* spectrometer hosts, address 2 is reserved for the HAL (unless the acquisition communication is done from a separate SCSI controller). On UNITY and VXR-S spectrometer hosts, SCSI address 3 is taken by the HAL and cannot be used.

[4] On VXR-S and UNITY spectrometers, SCSI address 3 is taken by the HAL. Only one SCSI disk can be mounted internally because the bottom drive automatically has address 3, so that the only internal drive must be moved to the upper position (using address 1).

[5] Remember that on VXR-S and UNITY spectrometer hosts, address 3 is taken by HAL and on UNITY*plus* spectrometer hosts, address 2 is taken by HAL. On these spectrometers, the total length of the SCSI chain is rather critical. Use short SCSI cables whenever possible. If you have problems accessing any SCSI device or the acquisition computer, a second SCSI controller to communicate with the HAL is recommended.

3. To test the SCSI addressing, use the monitor command `probe-scsi` (the output depends on the actual SCSI configuration):

   ```
   Type  'go' to resume
   Type  help  for more information
   ok probe-scsi
   Target 2
    unit 0   Disk   SEAGATE ST32550W SUN2.1G041400878689
                    Copyright (c) 1995 Seagate
                    All rights reserved ASA2
   Target 3
    unit 0   Disk   SEAGATE ST35660N SUN05350638DR630663
                    Copyright (c) 1993 Seagate
                    All rights reserved 0000
   Target 4
    unit 0   Removable Tape
                    ARCHIVE VIPER 150  21531-004 SUN-04.00.00
   Target 6
    unit 0   Removable Read Only Device
                    TOSHIBA XM-4101TASUNSLCD175506/24/95
   ```

   In the output you should recognize all devices that are currently connected to the SCSI bus and switched on, and you can immediately recognize problems in the case of SCSI address conflicts.

4. Use the `go` monitor command to continue:

   ```
   ok go
   ```

Hints on Adding SCSI Devices

Here are a few general hints about adding SCSI devices to your system.

- The `dmesg` command can show you which SCSI addresses are already taken. If the `dmesg` output is cluttered with error messages, you may use the monitor command `probe-scsi` instead, as shown in the previous section.
- Before starting the installation, shut down and switch the power off the workstation and all peripherals properly (see "Shutting Down the System," page 103).
- The newer Unipack desktop enclosures are self-terminating, but on workstations with older desktop enclosures, make sure the last device in the SCSI bus has a termination plug.
- Switch on the workstation, and then the peripherals. Abort the automatic boot process using Stop-a, and then boot the system using the `boot -r` command. If you can boot up and operate as before, the hardware installation is complete (for the software part of the installation, see "Adding a Disk (Software)," page 51).
- If the system does not boot, try these troubleshooting steps:
  a. Check for a bad SCSI connection.
  b. Make sure you don't have two SCSI devices using the same SCSI ID (change the SCSI address of the new disk drive and try rebooting with `boot -r`).
  c. Determine if you have exceeded the maximum cable length for your system (use shorter SCSI cables or add a second SCSI controller).
- If your system has free SBus slots and you have a shortage of free SCSI addresses or a problem with the SCSI cable and bus length, you can always add additional SCSI controllers.

Chapter 2. Software Installation

Sections in this chapter:

- 2.1 “Installing Solaris,” this page
- 2.2 “Installation and Customization of VNMR,” page 49
- 2.3 “Adding a Disk (Software),” page 51
- 2.4 “Adding Memory,” page 55
- 2.5 “Increasing Swap Space,” page 56
- 2.6 “Changing the Hostname,” page 57
- 2.7 “Adding in Parts of Solaris “After the Fact”,” page 58

The standard Solaris 2.x installation is covered in the manual *VNMR and Solaris Software Installation*. It still is worthwhile to focus on a few points around the installation of both Solaris 2.x and the VNMR software. The following sections are based on Solaris 2.5.1; most of it also applies to earlier Solaris 2.x releases, even though the installation tools have undergone considerable changes. Try to *understand* what the various steps are doing—then you will be able to also follow the following advices and hints even if the installation tools work differently on your version of Solaris.

2.1 Installing Solaris

To avoid future trouble, performance degradation, maybe even having to reload Solaris or VNMR at some point, it is a good idea to clarify a few more issues before starting the installation:

- What is the minimum swap space size for your computer? The optimum swap size?
- What Solaris software options should be loaded? “Developer System Support” or just “End User Support”? With only End User Support, you are not be able to compile C programs (even if you load the VNMR GNU option) such as pulse sequences (VNMR seqgen and psggen commands), or generate C program-based weighting functions (wtgen command). These are the primary limitations of the End User Support option we know of—there may be others. Note also that many processing facilities from the user library are C programs that may need compiling.
- Do you plan to load extra software (such as CDE, Wabi, Netscape, or other third-party software packages) on your computer? With VNMR 5.3 or later software, we definitely recommend loading CDE (Common Desktop Environment). With Solaris 2.6, CDE is included automatically.
- Do you plan adding extra disks in the near future?

Swap Space

Under SunOS 4.x and earlier Sun operating systems, the swap space was used to *duplicate and extend* the computer memory. Under Solaris 2.x, the swap space *only acts as the virtual extension* of the installed RAM. In theory (according to Sun), Solaris systems with enough RAM (64 MB or more) can even run without any swap space! On the other hand, running out of virtual memory can have pretty nasty consequences, such as the inability to start an acquisition or even start any new process. Also, programs operating on large data sets such as VNMR can be configured to use as much of the RAM as possible (e.g., by setting the environment variable `memsize`, defined in `~/.login`, to the number of megabytes of installed RAM[1]).

Certain programs, such as Adobe Acrobat Reader, tend to consume additional RAM, the longer they stay in use continuously. If you encounter such problems, you may want to periodically exit from such programs, to avoid excessive virtual memory consumption.

All this leads to the recommendation to adjust the size of the swap space to at least 100 MB, or twice the size of the installed RAM, whichever is higher. If you intend using several memory-consuming applications in parallel (e.g., under CDE, using the various desktops), you better increase the swap space beyond the recommended minimum. On the other hand, you don't need to plan ahead for future memory expansions because you can always add swap space later on (see "Increasing Swap Space," page 56).

Planning the Disk Layout for Solaris 2.5.1

Solaris 2.5 does a very good job at setting up the hard disk layout—but still, you can avoid disk clutter if you plan the installation carefully. If you follow the instructions in the standard manuals, everything should work fine, but it may be more difficult to add extra software packages you may find free disk space in the "wrong" disk partitions, and so forth. If you have multiple disks, you may even obtain improved system performance by selecting an appropriate disk layout.

For planning the disk layout, you primarily need to first collect some information:

- Try to get a rough idea about the size of the disks on your system. How many disks are there? are they 424 MB? 535 MB? 1.05 GB? 2.1 GB? Note that you can not install Solaris 2.5 and VNMR in a sensible configuration if your workstation is equipped with less than at least a single 424 MB disk. Solaris 2.5 requires at least 220 MB (350 MB in the configuration recommended for use with VNMR, up to 600 MB in a maximum configuration), the VNMR 5.3 software takes up between 60 and 120 MB of disk space (VNMR 6.1 can easily take up almost 200 MB). With a single 424-MB disk you will be very limited in disk space. Even with a 535-MB disk, you are restricted in the amount of software that you can load if you still want to be able to store a few spectra and do some 2D NMR transformations. Maybe this is the point where you should consider adding a second disk. If you install it at the same time as Solaris, this causes the least amount of extra work.
- Given the available disk space, you need to decide which Solaris options to load. If you want to be able to compile pulse sequences, you must select "Developer's

[1] The `memsize` variable defines how much memory is used by VNMR in 2D transformations. The default setting for this variable is 8, which allows performing a 1024 × 512 2D transformation "in core." Setting `memsize` to 32 raises this limit to 2048 × 1024, etc. For single-user systems, `memsize` should be adjusted to the number of megabytes of installed RAM. For multiuser systems, the sum of all memsize values used concurrently should not exceed the number of megabytes of installed RAM.

Support." Note that unless you reserve plenty of extra disk space in `/opt` and `/usr`, you will not be able to load extra options (such as upgrading from the End User Support to the Developer Support option) "after the fact." In most cases, you will need to reload UNIX if you find later that you are missing certain options.

If you have multiple disks, we recommend that you make the largest free partition `/export/home`. This becomes your working disk. 2D/3D acquisitions and processing always take up more disk space than expected. If there is free space on the main system disk, don't use that as `/export/home`, but mount it as `/data`.

Also, if you have `/export/home` on a separate disk, system performance will probably be better. A single disk has to do frequent and long seeks between the cylinder groups containing UNIX (`/usr`, `/dev`, `/tmp`, `/etc`) and the cylinder groups that contain VNMR and the user directories. With more disks, this principle can even be expanded.

Third-party software (other than VNMR) is usually loaded into `/opt`. You need to increase the size of this partition if you want to load such software. Solaris 2.5 (and 2.5.1) comes with two optional software packages, CDE and Wabi:

- CDE (Common Desktop Environment) has become the standard user interface in Solaris releases, replacing OpenLook[2]. CDE 1.1 (delivered with Solaris 2.5.1) is installed in `/usr/dt`. It requires 28 MB of disk space for a standard installation (CDE End User), or 47 MB if you plan writing and compiling software for the CDE platform (Developer and End User). If you want to also load the Answerbook package that provides extensive online documentation, you need to add an extra 120 MB to either of the above options.
- Wabi requires at least an extra 60 MB in `/opt/SUNWabi`. Wabi is a Windows emulation software and probably rarely used on systems running VNMR.

CDE and Wabi can also be installed in other partitions (it's just more work), so if you are not sure, you should not worry too much right now.

Table 5 gives you an idea of the (recommended) disk space consumption with various Solaris 2.5.1 software selections. The table indicates that, in theory, it is possible to install a very limited version of Solaris 2.5.1 on a 207-MB disk, but the remaining space is not sufficient for VNMR. Also, in the minimum configuration, the swap space is very limited, and compilations (such as pulse sequence compilation) are not possible. To enable compilations, we would need to load the developer configuration.

On the other hand, a system with a 207-MB disk could be used if `/export/home` (VNMR and all user data) is mounted from a remote host. Then, we could even increase the swap space to the recommended minimum and slightly increase `/var`, to have more disk space for printing and plotting spooling. In this smallest configuration, all partitions (except for `/var`) are full and leave less than 1 MB of free disk space. This means that files on these partitions will be more fragmented and that the disk performance will be worse. Whenever possible, specify "recommended" (rather than "minimum") partitions sizes—it helps performance.

Note that on spectrometer hosts, the root partition (`/`) should be made bigger, as outlined in the Table 5. We recommend making the root partition 25 MB on UNITY*plus*, UNITY, and VXR-S spectrometer hosts, and 30 MB for UNITY*INOVA*, *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* spectrometer hosts.

With all "End User" configurations, `/var` is left relatively small, because compilations cannot be done anyway. However, it is definitely recommended to *increase* `/var` even

[2] OpenWindows is still supported, but Sun is putting only limited development into it.

Table 5. Minimum Recommended Partition Sizes for Solaris 2.5.1 With CDE

File System	*Root (/)*							*Total*	
	Workstation	*Spectrometer*	*swap*	*/var*	*/opt*	*/usr*	*/usr/openwin*	*Workstation*	*Spectrometer*
End User, *(minimum,* ***not recommended****)*	13	30	32	10	9	45	54	163	180
End User, *(*`/export/home` *is remote)*	13	30	56	12	9	45	54	189	206
End User	14	30	64	30	11	53	64	246	262
End User + CDE	14	30	100	30	15	71	64	314	330
End User + CDE + Answerbook	14	30	100	30	15	191	64	424	440
Developer *(minimum,* ***not recommended****)*	13	30	64	30	9	82	110	308	325
Developer	14	30	100	60	20	96	129	419	435
Developer + CDE	14	30	100	60	20	143	129	466	482
Developer + CDE + Answerbook	14	30	100	60	20	263	129	586	602
Entire Distribution + CDE + Answerbook	17	34	100	60	60	277	146	660	680

without the extra space requirements for compilation. `/var` *is also used for print spooling*. With the developer's option, `/var` should be set to 50 to 60 MB, to allow for compilation.

It is not necessary to have `/usr/openwin` on a separate partition. This is just a possibility to move a portion of UNIX onto a second disk if necessary, such as on systems with a small first disk. If there is enough space to store the software on a single disk (if the system disk is 1 GB or more), we would rather recommend *not* to set up a separate partition for `/usr/openwin`. In this case, the disk space requirement for `/usr/openwin` indicated above needs to be added to `/usr`.

The `/opt` slice is used for extra software (such as Wabi or many third-party software packages, such as FrameMaker). If you want to also install Wabi, you should *increase* `/opt` *by* 60 MB. Note that for Solaris versions prior to 2.5.1 (namely 2.5, 2,4, and 2.3) the `/opt` slice needed to be made substantially bigger than listed above, in order to be able to load the Solaris patches. See the manual *VNMR and Solaris Software Installation* for more information.

In principle, even `/opt`, `/var`, and `/usr` don't need to be installed on separate slices—it is even possible to *install Solaris in a single disk slice.* This has several advantages:

- All the free disk space is available in a single pool, which allows for more flexibility. In a multiple-partition setup. lots of free space is locked away in various partitions and remains mostly unused.
- With the modern backup media (tapes with capacities of 2.5 to 14 GB), even a level 0 dump of an entire disk normally fits onto a single tape; therefore, the entire system can be backed up in one step, greatly simplifying this task. In a multiple-partition setup, every slice must be backed up individually (using `ufsdump`).

On the other hand, the partitioned setup has some advantages:

- `/var` contains the spooling directories for print services plus directories for temporary files, as they are created during a compilation. There will be a fair amount of I/O on this slice. If you incidentally overfill the spooling area, UNIX will (with the exception of printing, of course) still be fully functional.

- `/usr` contains virtually all of the standard UNIX software, including the user interface, etc. Whenever possible, this slice should be left alone after the installation—you can even consider mounting it read-only.
- Consider the case where your system somehow loses a disk slice (due to file system corruption, a head crash, etc.). If a UNIX slice is affected, you could reload UNIX (or eventually restore single files from a backup) and still keep your data. And if a user slice is affected, you could restore that from backups with little effort. Alternatively, if you have a single disk slice when something goes wrong, you have lost both Solaris *and* your data. Restoring that from a backup tape can be tricky (remember, you have just lost UNIX), and you end up reloading Solaris and restoring *all* your data from a backup.

Up to VNMR 6.1A (and Solaris releases up to Solaris 2.5.1), we have followed the official Sun preference and recommended a multiple-slice installation. With Solaris 2.6, Sun is changing this policy, which we follow with our recommendation, see below.

It is unlikely that you will need the entire distribution, but we recommend loading CDE, which has become the standard user interface on Sun workstations starting with Solaris 2.6. VNMR 5.2F and later versions are CDE-compatible; VNMR 5.1 can be installed on a system with CDE, but it is recommended *not* to select CDE when using VNMR 5.1. The Sun Answerbook is a set of comprehensive online documentation and may save you from having to purchase extra books.

Note that for the installation of some Solaris patches, patch clusters in particular, you must leave at least 4 MB of free space in each of the `/`, `/var`, and `/opt` slices, and at least 15 MB of free space in `/usr`.

Planning the Disk Layout for Solaris 2.6

For Solaris 2.6, the disk space requirements have increased compared to Solaris 2.5.1 (see Table 6), but the installation is greatly simplified because in Solaris 2.6 CDE is part of the standard software distribution. We don't need to reserve disk space for CDE and then install that software from an extra CD-ROM, using an extra installation step. On the other hand, installing Solaris 2.6 involves several CD-ROMs.

Table 6. Minimum Recommended Partition Sizes with Solaris 2.6

File System	*Root (/)*							***Total***	
	Workstation	*Spectrometer*	*swap*	*/var*	*/opt*	*/usr*	*/usr/openwin*	*Workstation*	*Spectrometer*
End User, *(minimum, **not recommended**)*	24	40	32	10	8	135	77	286	302
End User	24	40	128	30	11	155	91	439	455
Developer *(minimum, **not recommended**)*	24	40	64	30	11	232	236	597	613
Developer	24	40	128	60	25	266	270	773	789
Entire Distribution	25	42	128	60	60	328	391	992	1008

According to our recommendations, Solaris 2.6 requires 439 MB of disk space (End User System Support), 773 MB (Developer System Support), or even close to 1 GB (Entire Distribution). For "real world" installations involving VNMR (on a system with 64 MB of RAM), we recommend the slice sizes listed in Table 6. For spectrometer hosts, and additional 16 MB are required in the root slice. `/usr/openwin` may be kept separate if the main disk is less than 1 GB. If your system is equipped with less than 1 GB of disk space, we strongly recommend adding another disk.

The `/usr` and `/usr/openwin` slice sizes are those proposed by the Sun installation software. It turns out that these slices contain a lot of unused disk space reserves (100 to 150 MB). As of Solaris 2.6, we recommend installing Solaris in a single disk slice. Sun obviously rates the advantages of this solution more important than the few disadvantages (see "Planning the Disk Layout for Solaris 2.5.1," page 40).

The term "single-slice installation" is actually incorrect, because you always have the swap space, which includes `/tmp`, in a separate slice. Therefore, in this simplified scheme you end up with two slices:

- *Main software and user slice* – The developer version of Solaris 2.6 takes up about 400 MB, compared to about 660 MB in a multiple-slice installation (see Table 6).
- *Swap space* – Typically set to about 128 MB for a RAM size of 64 MB.

VNMR 6.1 adds close to 200 MB to the software and user slice, but even then, there should be *about 250 MB of free space* on systems with only a single 1.05-GB disk. A single 535-MB disk is definitely not enough for Solaris and VNMR, even with only the End User option and a single-slice installation.

Adjusting the Disk Layout, Installing Solaris 2.5.1

Let's walk through an installation—this gives us an opportunity to discuss the essential points. Our sample system is a networked SPARCstation 4 with internal 535-MB disk and an additional external 2.1-GB disk. We want to load the developer configuration, including CDE and Answerbook.

Note that the numbers shown for the disk slice sizes were adjusted for this particular example (disk size and software selection) and may not be appropriate for your system. Refer to the Table 5 and Table 6 for recommended minimum slice sizes under Solaris 2.5/2.5.1 and 2.6, respectively.

For the installation, we shut the system down, insert the Solaris CD-ROM, and enter:

```
ok boot cdrom
```

After a while, we get an OpenLook user interface, and we go through the initial questions:

- The hostname. We recommend using lowercase alphanumerics only (minus signs are allowed also, but only inside the name). Start with an alphabetic character and don't make it longer than eight characters total.
- We declare the workstation as "Networked",
- We enter the (numeric) IP address (such as 193.72.128.2, as provided by the local network administrator).
- Then, after confirming these entries, we select the naming service. Most systems use "None" (see Chapter 17, "Networking via Ethernet," for NIS and NIS+). The installation software asks for a confirmation on this information.
- We then specify whether the local network is using subnets or not. If yes, we specify the subnet mask.
- Now we select the time zone for the system and set the current local time. Again, we confirm this information.

Now, we come to the software pages:

- Before we can select a system type, we may get a screen that tells us that the system can be upgraded. The “upgrade” option only makes sense if we *know* that the current disk layout will be okay for the new software selection. In the vast majority of the cases, we choose “Initial”, which simply erases the current software and replaces it with a new installation.
- Now, we specify the system as “Standalone” (the other two cases are not covered in this or other VNMR manuals).
- We now select “Developer System Support.”

  If we wanted to, we could now use the “Customize” button to change details in the current software selection. Unless you have a distinct reason for doing so, *don’t*. (In earlier Solaris releases, this could even lead to installation errors if the partition sizes were not adjusted carefully). In almost all cases, we select “Continue” directly.
- On the next page, we select the disks for the installation. It is recommended *to select all disks, even if selected partitions are to be preserved. If you don ‘t select a disk, it is relabeled anyway, and its contents are lost*.
- The next window asks whether we want to preserve existing data. This is only possible. if we have partitions (typically on separate disks) that will not change in size and that we want to be preserved. If this is the case, select “Preserve” and (in a new window) specify which partitions need to be preserved. After this, we again select “Continue”.
- The next page asks us whether we want to repeat (if Solaris was already installed before) or do an “Auto-layout.” Selecting Auto-layout changes the current partition map, and during the installation, the partitions that change in size are erased and re-created, except for partitions specified as “to be preserved”. We select “Repeat Auto.”
- On the next window, we specify which slices we want to create. “/,” “swap,” and “/usr” are selected by default. We also select “/opt,” “/var,” and “/usr/openwin” (if necessary, we could still eliminate any of these extra slices in the next step).
- After “Continue,” we get an overview over the current partition map (on all selected disks). Because we want to fine-tune the partition map and make sure that we can install the CDE and Answerbook software packages afterwards (from a separate CD-ROM), we select “Customize.”

The installation software determined the recommended (and minimum) sizes for the various partitions under the current software selection. It then figured out that the remaining space on the first disk was too small for `/export/home` with reasonable size, and therefore it decided to mount the second disk as `/export/home` (this fits nicely with our recommendation). Still, on the first disk there was some free space left. All partitions have been increased proportionally until all the disk space was used up. The following partition map is presented:

	Disk: c0t3d0 510 MB			Disk: c0t2d0 2028 MB	
0	/	18	0		
1	swap	36	1		
2	overlap	510	2	overlap	2028
3	/var	114	3		
4	/usr/openwin	129	4		
5	/opt	114	5		
6	/usr	98	6		
7			7	/export/home	2028

Capacity:	510 MB	Capacity:	2028 MB
Allocated:	510 MB	Allocated:	2028 MB
Free:	0 MB	Free:	0 MB

We can click on any of the partition sizes, and the window indicates the minimum and recommended sizes for that partition. If we want to increase a partition size, we first must create some free space by decreasing the size of an other partition (just make sure you restore at least the minimum size again at the end). We can also eliminate mount points (by erasing the name from the second column and setting the slice size to zero) and create new ones, as required.

Based on the required partition sizes given in Table 5 for a spectrometer configuration with CDE and Answerbook, we decided to move `/usr/openwin` to the second disk. This allows the rest of Solaris to reside on the 535-MB disk, along with 128 MB of swap space.

The target partition map looks as follows:

	Disk: c0t3d0 510 MB			Disk: c0t2d0 2028 MB	
0	/	30	0		
1	swap	128	1		
2	overlap	510	2	overlap	2028
3	/var	60	3		
4			4	/usr/openwin	130
5	/opt	20	5		
6	/usr	271	6		
7			7	/export/home	1898
	Capacity:	510 MB		Capacity:	2028 MB
	Allocated:	509 MB		Allocated:	2028 MB
	Free:	0 MB		Free:	0 MB
	Rounding Error:	1 MB			

To change the original slice map, we proceed in steps:

- Eliminate `/usr/openwin` from the first disk.
- Increase the root partition to 30 MB.
- Change `/var` to 60 MB.
- Increase the swap space to 128 MB (see "Swap Space," page 40).
- Set `/opt` to 20 MB (note that this does not allow for much software).

For the Developer Support option with CDE and Answerbook, `/usr` has a recommended size of 263 MB. The remaining disk space allows increasing `/usr` to 271 MB, which is fine because the recommended size already leaves some free space in that slice. Make sure you use all the available free space—any "Free" disk space not allocated to a partition is unusable after the installation. Therefore, we do the following:

- Allocate the remaining 271 MB to `/usr`.
- Adjust the second-disk partition map by reducing `/export/home` by 130 MB.
- Add `/usr/openwin` on any of the unused slices of the second disk and set the size to 130 MB.

The "overlap" partitions are disk slices covering the entire disk and can be used when a disk is used as single slice.

If there is substantial free space on the first disk, the installation software proposes to mount this as `/export/home`, and a free external disk (partition) is then mounted as `/export/home0` (further disks as `/export/home1`, etc.). As discussed previously, we recommend mounting the largest free partition as `/export/home` and mounting other free partitions as `/data`, `/data0`, etc. To do this, change the mount point `/export/home` to `/data`, and then change `/export/home0` to `/export/home` (changing `/export/home0` first does not work, because this would temporarily lead to duplicate mount points).

When we are done with the partition map, we select "OK", which returns us to the "File System and Disk Layout" overview, in which we select "Continue." We can now set remote mount points (if there is another system with file systems that are exported). For each one we need to specify the host name and the IP address of the remote server, the remote file system, and the local mount point (see Chapter 17, "Networking via Ethernet," for more information). We can also do this later if we want.

After selecting "Continue," the software presents an installation profile, summarizing all our input and selections. Up to this point, we can still select "Exit" or abort the installation (Stop-a), and the disks would remain untouched. This gives us a chance to try out the installation tool and gain experience with it, without having to be afraid of consequences, in case we make a mistake.

After we select "Begin Installation," we get a last prompt, asking whether we want to reboot after the installation or not. Because we need to reboot anyway after the installation, we select "Reboot after Installation," and then the Solaris installation starts.

After it completes, the system reboots and prompts for a root password. Once that is defined, it completes the bootup, giving us a `login` prompt. We log in as `root` and eject the CD-ROM by entering:

```
# eject cdrom
```

We can now insert the CD-ROM labeled "Desktop 1.1" and follow the instructions given in the *Solaris Common Desktop Environment: Installation and System Administration Guide* that comes with the software. We enter:

```
# cd /cdrom
# ls
cdrom0    solaris_2_5_1_desktop_1_1
# cd cdrom0
# ls
CDE         Copyright   ODBC2.11   Wabi
# cd CDE
# ls
Copyright   PowerPC     sparc      x86
# cd sparc
# ./install-cde
```

This presents us with the options:

1. Start the installation
2. Modify the configuration settings
3. Cancel the installation.

We choose to modify the configuration settings because, by default, only the End User package and the "Solaris Desktop Login at System Boot" are selected.

On the next screen, we select the "Developer CDE Package"(3) and "Answerbook CDE Package" (4). For each of these, we enter "y" (or "YES", lower- or upper-case) to activate the item (only if we want these options, of course). If we want or need to install CDE into a different location (the default is `/usr/dt`), we can enter an alternate file path by selecting “1”.

When we are finished, we return to the main menu and start the installation.

Note: If you understand the above section, you should also be able to install Solaris 2.6 (which is much easier to install). Therefore, we don’t discuss this separately here. We recommend installing Solaris 2.6 in a single disk slice, in which case all the steps for above setting up a proper disk partitioning are unnecessary.

Solaris Patch Installation

Particularly for UNITY*INOVA* systems, VNMR requires certain Solaris patches to be installed, for example, the patch 103582-xx (TCP patch) for Solaris 2.5.1. There are Solaris patches on the VNMR CD-ROM (together with installation instructions in the manual *VNMR and Solaris Software Installation*). Before you start installing, consider the following points:

- Sun is constantly updating their patches. The patches on the VNMR CD-ROM may be outdated already when you receive the VNMR software. You should check the SunSolve Web site (`http://sunsolve.sun.com/`) or the Sun patch FTP server (`ftp://sunsolve.sun.com/pub/patches/patches.html`) for newer versions of the required patches. In the patch number, the two digits after the dash are the patch version number.
- Even within a given Solaris release, there may be newer versions (such as Solaris 2.5.1 “Hardware: 4/97”, Solaris 2.5.1 “Hardware: 8/97”, or Solaris 2.5.1 “Hardware: 11/97”) that automatically include and install many patches, including those required by VNMR. To check whether a particular patch (such as the TCP patch 103582-xx mentioned above) is already installed, use the following command (no output means that the patch is not installed):
  ```
  showrev -p | grep 103582
  ```
- Apart from the individual patches, Sun also offers entire clusters of recommended patches, such as `2.5.1_Recommended.tar.Z` (typically files of 20 to 30 MB after compression), that permit bringing your Solaris installation to the latest level. Note, however, that the installation of patch clusters requires extra disk space. If don’t have at least 36 MB of free space in `/var` (`/var/sadm`), you need to use the “`-nosave`” option for the cluster installation, i.e., you will not be able to back out individual patches in the case of problems (in practice—at least on a typical VNMR system—this is rarely done anyway). Under Solaris 2.5.1, a “nosave” installation is achieved with
  ```
  cd 2.5.1_recommended
  ./install_cluster -nosave
  ```
  Installation of a Solaris 2.x patch cluster takes 1 to 2 hours.

Assigning a Password to the System Administrator

The UNIX system administrator’s login name of `root` is given permission by UNIX to access all the files. With this permission, `root` can erase files, change files, move files, and perform many other functions on all system files and directories. Therefore, it is extremely important that the access to the system as `root` be controlled.

A simple and effective way of controlling access is by assigning a password to `root`. Solaris asks for a `root` password when installed. Instructions are given at that time. On pre-Solaris systems, the password is assigned by using the `passwd` command in UNIX.

The UNIX system administrator should log in as `root`, then type the `passwd` command, which will prompt the administrator through the password assignment. Some of the requirements for a password include:

- The password must be more than six letters and numbers.
- The password cannot contain certain characters that have a special meaning to UNIX. Some of these characters are * / \ ? % & |. Consult Sun Microsystems manuals for a complete list of reserved characters or open a terminal tool or command tool window and type `man passwd`.

When a `root` password has been assigned, it should be kept private. Widespread knowledge of the password negates the protection. The password should be kept in a safe place so that it is available in the event that service on the unit is required. It is extremely important to understand that `root` *must* enter the correct password to gain access to the system as the superuser. If the `root` password is lost or forgotten, there is *no* way available to learn the password from the system. For systems running Solaris, even booting up as a single user requires the `root` password.

2.2 Installation and Customization of VNMR

After loading the VNMR software, the output devices (printers, plotters, etc.) need to be defined (see Chapter 4, "Output Devices, Terminals, Serial Ports," for further information). You also must configure VNMR using `config` (in VNMR); otherwise, it does not function properly. On standalone workstations, it is most convenient to set all configuration parameters to the same values as on the spectrometer that the workstation mostly relates to (except that you should not declare the workstation as a spectrometer).

Backup your files `devicenames`, `conpar`, and `shims` from `/vnmr` (along with any other file that you have customized) onto tape. These are the most important customized files within `/vnmr`. When reloading the complete software, you can then read back those files into `/vnmr` (this only applies when reloading the *same* software, of course; if you are loading a new version of VNMR, you can still load those backup files into a temporary directory for comparison purposes, but you should not reuse old `conpar` or other files (other than shim settings) because these could be incompatible with a new VNMR release.

Installing New Users

Installing new users is described in the manual *VNMR and Solaris Software Installation*. Unless you have very large disks, be careful not to create too many users. Each user requires a series of experiments (sometimes large experiments) as well as saved NMR data in subdirectories. Your free disk space can vanish within an incredibly short period of time.

Try to group the users into a few shared home directories. It will never be easy to keep file system discipline with many users unless `root` vigorously cleans up all the home directories every now and then.

Assigning New Users

One of the most important tasks of the UNIX system administrator is assigning new users. The manual *VNMR and Solaris Software Installation* covers in detail how to handle this responsibility. Each user in a UNIX system requires supporting directories and files to allow the user to enter the environment.

The VNMR software includes the UNIX command `makeuser` that sets up a new user's entry in the `passwd`, `shadow` (Solaris only), and `group` files, as well as makes a home directory and all the subdirectories required by VNMR. The creation and modification of these directories and files is the task of the UNIX system administrator.

Password File

To log on to the system, each user must have an entry in the password file `/etc/passwd`. On Solaris systems, the entry for a user is similar to the following:

```
john:x:77:30:John Smith:/export/home/chem/john:/bin/csh
```

On pre-Solaris systems, the entry for a user is slightly different:

```
john:8nmdft63&:77:30:John Smith:/home/chem/john:/bin/csh
```

Both entries have the following information (reading from left to right):

- Field 1 is the user's login name. In the entries above, john is the login name.
- Field 2 is the letter x on Solaris systems or the user's encrypted password on other systems. Solaris uses another file, `/etc/shadow`, to hold the encrypted password and other information about the user. Only `root` can even view the `shadow` file.
- Field 3 is the user's ID number. In the entries, the user ID is 77.
- Field 4 is the user's group number. In the entries above, the group number is 30.
- Field 5 is the user's title, telephone number, etc. This information is used by the UNIX commands `whois` and `finger`.
- Field 6 is the user's home or login directory. Note that Solaris uses the `export` directory in the path.
- Field 7 is the user's shell (the type of command interface to UNIX). In the entries above, the entry is `/bin/csh` for the C shell. Another shell is `/bin/ksh` for the Korn shell. No entry in this field indicates a Bourne shell.

Refer to Sun Microsystems manuals or reference books on UNIX and Solaris for further information about password files.

Group File

The group file (`/etc/group`) contains the names of groups that exist in the system and the users that are members of each group. This file is important in that it sets the access to files not owned by the user. An entry in the group file may look like:

```
nmr::30:vnmr1,mike,jim
```

The information in this entry shows that the nmr group number is 30 and it has `vnmr1`, `mike`, and `jim` as members.

Home Directory

In a situation where the individual users are not familiar with UNIX, it may be necessary for the UNIX system administrator to assist in personalizing the user's home directory. In particular, there are "dot" files (files with a period as the first character in the name) that can be edited to provide shell commands personalized for the user. Such files include the following:

- `.cshrc` is the C-shell run command file.
- `.login` is the C-shell home directory command file.
- `.dtprofile` is used by CDE for initialization.

- `.dt` is the directory that contains CDE files.

Files starting with a dot are called hidden because they are not displayed by the usual UNIX `lf` or `ls` commands (or the VNMR `lf`, `ls`, or `dir` commands); however, they are displayed if the UNIX `ls -a` command is used (or `ls('-a')` in VNMR). See Chapter 12, "File Security, Repair, and Archiving," for more information.

Sending System Messages

System messages are another responsibility of the UNIX system administrator:

- The message of the day, stored in the file `/etc/motd`, can be used to remind users of meetings, preventative maintenance, etc. `/etc/motd` is displayed every time a user logs in; however, the message may not show very long because it is overwritten by the windows that appear as part of the process of starting VNMR. A better way to pass information to users is to use instead `/vnmr/bootup_message`. This file is displayed in the text window every time VNMR is started.
- When shutting down a multiuser system, the `shutdown` command can be used to send messages to each active user to save files and log off. Only the UNIX system administrator should invoke the `shutdown` command and then only with ample warning if possible (obviously, a single-user system can be shut down at any time without system messages). For more information, see "Shutting Down the System," page 103.

2.3 Adding a Disk (Software)

The instructions in this section should enable the user to add a second (or third) hard disk to the system without having to reload any software (if this has to be done anyway, the installation can proceed as described above). It turns out that installing a second disk by software is rather easy under Solaris because the formatting routines can be called from within the standard Solaris environment. The total time involved should be about 30 minutes plus the time it takes to format the drive.

1. Start by shutting down UNIX, switching off the Sun and installing the hardware as described in the hardware documentation and in "Adding a Disk (Hardware)," page 35.
2. Once the disk has been installed, reboot UNIX again. It is important that you reboot the system with the `boot -r` option. If necessary, interrupt the automatic boot process using Stop-a or, alternatively (safer), let the system boot up automatically, and then log in as `root` and bring the system down to monitor level using
   ```
   # init 0
   ```
   When you get the ok prompt, type
   ```
   ok boot -r
   ```
   This causes the system to create the necessary new entries in `/devices` and `/dev`.
3. Log in as `root`.

   If the system does not boot at all, either you have a bad SCSI connection or you have created a SCSI address conflict. Change the SCSI address on the new disk to an unused number (between 0 and 3), check the connections, and try again. If the problems persist, use shorter SCSI cables (the SCSI bus may be electrically overloaded[3]).

4. If you want to format the new disk (reformatting is optional because most disks are already formatted), enter the `format` command. It presents you with a list of available disks:

```
# format
Searching for disks...done

AVAILABLE DISK SELECTIONS:
  0. c0t2d0 <SUN2.1G cyl 2733 alt 2 hd 19 sec 80
    /iommu@0,10000000/sbus@0,10001000/espdma@4,8400000/
                                   esp@4,8800000/sd@2,0
  1. c0t3d0 <SUN0535 cyl 1866 alt 2 hd 7 sec 80
    /iommu@0,10000000/sbus@0,10001000/espdma@4,8400000/
                                   esp@4,8800000/sd@3,0
Specify disk (enter its number): 0
selecting c0t2d0
[disk formatted]
Warning: Current disk has mounted partitions.

FORMAT MENU
    disk       - select a disk
    type       - select (define) a disk type
    partition  - select (define) a partition table
    current    - describe the current disk
    format     - format and analyze the disk
    repair     - repair a defective sector
    label      - write label to the disk
    analyze    - surface analysis
    defect     - defect list management
    backup     - search for backup labels
    verify     - read and display labels
    save       - save new disk/partition definitions
    inquiry    - show vendor, product and revision
    volname    - set 8-character volume name
    quit
format> q
#
```

Disks are presented in the order of the SCSI address: the “first” (internal) disk has SCSI address 3 (unless this is a spectrometer host for a VXR-S or UNITY) and is therefore be listed *last* (see Chapter 6, “Disk Organization,” for details).

In this particular case, a disk with mounted partitions was selected, which caused `format` to issue a warning. Of course, we don’t want to format such a disk and should quit again by entering `q` (commands in `format` don’t need to be typed completely; it is sufficient to type the first character(s) of a command name, such that the string is unique and the name can be recognized among the available options).

5. In case the disk label is unreadable or faulty, you can now create a proper disk label (for original Sun drives you can normally skip this step):

```
format> type
...
```

6. You should now select the proper disk type by selecting a number from the options given. If the disk is not an original Sun drive, it may be a nonstandard type. Therefore, you would have to specify “other,” which causes the program to ask for

[3] On UNITY*plus*, UNITY and VXR-S spectrometers, where the acquisition computer is accessed via SCSI bus, the restrictions in terms of SCSI load and maximum cable length are much tighter—apart from the fact that one SCSI address is taken by the host-to-acquisition link. In these cases, it may be advisable or necessary to use a second SCSI adapter (SBus card).

more information about the new drive (number of heads, number of tracks, number of sectors per track, interleave factor, and rotation speed). This information has to be extracted from the hardware documentation for the new disk.

7. Make a backup of the defect list for the disk using the `defect` program. You don't need a tape for this operation because you have the first disk available to temporarily store these data:

```
format> defect
...
defect> original
defect> print
defect> dump
Enter name of defect file: /newdisk.defect
defect> quit
Do you wish to commit changes to the current defect list? y
```

8. Continue with the formatting program. If you want, you could change the number of surface analysis passes now: the default is 2. If you have several hours, you can increase the default to 5, or more for overnight formatting; otherwise, you can skip the following steps:

```
format> analyze
analyze> setup
analyze entire disk [yes]? y
loop continuously [no]? n
enter number of passes [2]? 5
```

   All other lines can be answered by pressing Return (selects default).

```
analyze> quit
```

9. You can now proceed with the format program:

```
format> format
Ready to format. Formatting cannot be interrupted and   \
   takes x minutes (estimated). Continue? y
Formatting... done.
Verifying media...
...
```

10. Use the `label` program to label the disk:

```
format> label
Ready to label disk, continue? y
```

11. Before you quit the formatting program, you should change the partitioning of the new disk with the `partition` subprogram such that you can use the disk in a single, contiguous block. To do this, you need to adjust all partitions to zero size, except for partition 2 (the “overlap” or backup partition covering the entire disk, which we leave untouched) and partition 7, which we make the same size as partition 2— namely, the full disk. For a Sun 2.1-GB disk, we proceed as shown below (the starting partition map shown here is the default partition map for Sun 2.1-GB disks).

    The `partition` subprogram first presents us with a menu, which you can always recall by typing “?”. We start asking for a printout of the current partition map, and then we change all the partitions that need to be altered:

```
format> partition

PARTITION MENU
    0      - change ‘0’ partition
    1      - change ‘1’ partition
    2      - change ‘2’ partition
    3      - change ‘3’ partition
    4      - change ‘4’ partition
    5      - change ‘5’ partition
    6      - change ‘6’ partition
    7      - change ‘7’ partition
    select - select a predefined table
    modify - modify a predefined partition table
    name   - name the current table
    print  - display the current table
    label  - write partition map and label to the disk
    quit
partition> p
Current partition table (original):
Total disk cylinders available: 2733 + 2 (reserved cylinders)

Part   Tag   Flag   Cylinders   Size      Blocks
 0     root   wm     0 - 40     30.43MB   (41/0/0)    62320
 1     swap   wu    41 - 170    96.48MB  (130/0/0)   197600
 2     backup  wu    0 - 2732   1.98GB  (2733/0/0)  4154160
 3 unassigned  wm    0            0        (0/0/0)        0
 4 unassigned  wm    0            0        (0/0/0)        0
 5 unassigned  wm    0            0        (0/0/0)        0
 6     usr     wm    171 - 2732  1.86GB (2562/0/0)  3894240
 7 unassigned  wm    0            0        (0/0/0)        0

partition> 0
Part    Tag    Flag  Cylinders    Size      Blocks
 0      root   wm     0 - 40      30.43MB   (41/0/0) 62320
Enter partition id tag[root]: unassigned
Enter partition permission flags[wm]:
Enter new starting cyl[0]:
Enter partition size[62320b, 41c, 30.43mb]: 0
partition> 1
Part    Tag    Flag  Cylinders    Size      Blocks
 1      swap   wu    41 - 170     96.48MB  (130/0/0) 197600
Enter partition id tag[swap]: unassigned
Enter partition permission flags[wu]: wm
Enter new starting cyl[41]: 0
Enter partition size[197600b, 130c, 6.48MB]: 0
partition> 6
Part    Tag    Flag  Cylinders    Size      Blocks
 6      usr    wm   171 - 2732    1.86GB (2562/0/0) 3894240
Enter partition id tag[usr]: unassigned
Enter partition permission flags[wm]:
Enter new starting cyl[171]: 0
Enter partition size[3894240b, 2562c, 1901.48mb]: 0
partition> 7
Part    Tag    Flag  Cylinders    Size      Blocks
 7 unassigned    wm  0            0        (0/0/0)    0
Enter partition id tag[unassigned]:
Enter partition permission flags[wm]:
Enter new starting cyl[0]:
Enter partition size[0b, 0c, 0.00mb]: 2733c
partition> p
Current partition table (unnamed):
Total disk cylinders available: 2733 + 2 (reserved cylinders)

Part    Tag    Flag  Cylinders    Size      Blocks
 0 unassigned  wm    0            0          (0/0/0)     0
 1 unassigned  wm    0            0          (0/0/0)     0
 2     backup  wu    0 - 2732     1.98GB  (2733/0/0)  4154160
 3 unassigned  wm    0            0          (0/0/0)     0
 4 unassigned  wm    0            0          (0/0/0)     0
 5 unassigned  wm    0            0          (0/0/0)     0
 6 unassigned  wm    0            0          (0/0/0)     0
 7 unassigned  wm    0 - 2732     1.98GB  (2733/0/0)  4154160

partition> label
Ready to label disk, continue? y
```

```
partition> q
format> q
(...)
#
```

This process is mostly self-explanatory. `wm` in the `Flag` column stands for “write mostly”; this value is not relevant for partitions of zero size.

Partitions always start with a new cylinder. Therefore, it is best to specify partition sizes in full cylinders (e.g.: `2733c`). This avoids unused sectors at the end of a partition. Make sure you label the disk at the end. Only this action writes the new partition map to the disk.

12. The disk is now formatted and partitioned, but the software still doesn’t use it, of course. First, we have to create a UNIX file system in the new partition, and then we have to create a mount point (an empty directory, preferably in the directory `/`):

    ```
    newfs /dev/rdsk/c0t2d0ds7
    mkdir /data
    ```

13. Tell the system to mount this new partition automatically when booting up by adding the following line to the file `/etc/vfstab` (enter `vi /etc/vfstab` to use `vi` as a text editor):

    ```
    /dev/dsk/c0t2d0s7 /dev/rdsk/c0t2d0s7 /data ufs 3 yes -
    ```

14. We can now directly mount the new disk, and then move into it and check whether it is accessible:

    ```
    mount /data
    cd /data
    pwd
    ```

15. If the system is on a network, and the new disk and file system should be mounted from other systems, we should add an entry for the new file system `/data` (on a new line) to the text file `/etc/dfs/sharetab`:

    ```
    share -F nfs -o rw /data
    ```

16. To activate the file system sharing on that disk, we need to either reboot the system or call the command `shareall`; otherwise, NFS mounting is not possible with the new disk. See Chapter 17, “Networking via Ethernet,” for more details.

17. We can now transfer files (*`file1`*, *`file2`*, etc.) from `/home` to `/data`, using the `tar` command:

    ```
    cd /export/home
    tar cf - file1 file2 ... | (cd /data; tar xfBp -)
    ```

18. Now is a good time to reboot the system (make sure the disk check shows no errors):

    ```
    reboot
    ```

19. Log in again as `root` and check for files in `/export/home`. Use the `rm` command to remove duplicate files:

    ```
    cd /export/home; ls
    rm -r file1 file2 ...
    ```

20. If the data files still need to be accessible as if they were on `/export/home`, you can create a symbolic link for each of these files, for example:

    ```
    ln -s /data/file1 /export/home
    ```

2.4 Adding Memory

Memory boards normally come with installation instructions (the warnings about antistatic protection should be taken seriously). Adding RAM memory to the CPU does not normally require changing or installing any software, but you may want to adjust the size of the swap space.

For optimum 2D performance with VNMR on a single-user system, you may want to adjust the environment variable `memsize`, which is defined in each user's `~/.login` file, to the size in MB of the installed RAM memory.

2.5 Increasing Swap Space

If the swap space turns out to be too small, you may experience performance losses or more serious problems (such as the inability to start programs, run an acquisition in VNMR, etc.). We recommend adjusting the swap space to at least 100 MB or twice the size of the installed RAM, whichever is larger. This should do for VNMR, but there may be other software requiring more virtual memory (there are third-party applications requiring as much as 1 GB of swap space and virtual memory) or you may be using many applications in parallel (e.g., using the various CDE desktops).

If you suspect that your system suffers from swap space limitations, enter the command `swap –s` to display the current swap space usage:

```
# swap -s
total: 19032k bytes allocated + 5740k reserved =   \
     24772k used, 82096k available
```

Adding "external" swap space is done by creating a contiguous disk file in any disk partition (such as `/export/home` or wherever there is enough free space). The size of the additional swap space can be selected freely. Usually we add multiples of 4 MB.

1. Create the disk file that will serve as swap space extension, for example:
   ```
   mkfile 64m /export/home/swap
   ```
   This creates a 64-MB swap file in `/export/home`.
2. To activate the new swap space immediately, call
   ```
   # swap -a /export/home/swap
   ```
3. To verify the presence of the additional swap space, use `swap` with different options:
   ```
   # swap -s
   total: 18724k bytes allocated + 5732k reserved =   \
      24456k used, 115144k available
   # swap -l
   swapfile             dev  swaplo blocks   free
   /dev/dsk/c0t3d0s1    32,25      8 131592 131592
   /export/home/swap      -        8 131592 131592
   ```
4. To make this swap space addition permanent add a line to `/etc/vfstab`:
   ```
   /export/home/swap - - swap - no -
   ```

To disable this additional swap device (because you need the space on the disk), you can comment out the above line in `/etc/vfstab` (insert a pound sign "#" at the start of the line), reboot the system, and then delete the file `/export/home/swap`.

It is also possible to add swap space through Ethernet network (from a remote disk), but because the speed of swapping is essential to the overall performance of the system, this is not a recommended practice. It may also add considerable load to the network.

2.6 Changing the Hostname

The safest and easiest way to change the host name is to use the `sys-unconfig` command.

1. Enter:
   `sys-unconfig`
   The command issues a number of warnings and then “unconfigures” the system completely.

2. Reboot the system. You are asked to enter the relevant system and networking information, similar to when you install Solaris.

   You just need to be aware of the fact that `sys-unconfig` erases and initializes all network configuration files, such as `/etc/hosts`, and you need to restore any extra information in these files, such as remote hosts, after rebooting.

You can also change the hostname “by hand”, by modifying the appropriate system administration files. It’s not difficult, but it requires rebooting the system and should *not* just be done “on the fly.” Fortunately, the hostname is not stored in too many files.

1. Before you change the hostname locally, call the following command to unmount all (NFS-type) remote files:
   `umountall -F nfs`

2. Do the same command on all other systems that mount file systems on local disks.

3. Change the `hostname` locally in the following ASCII files:
   ```
   /etc/hostname.xx0
   /etc/hosts
   /etc/hosts.equiv
   /etc/nodename
   ```
   On networked systems, the file `hostname.xx0` is named either `hostname.le0` or `hostname.le1` (or `hostname.ie0`, or `hostname.ie1`).

4. You may want to change the current host name with the `hostname` command:
   `hostname` ***`new_hostname`***

5. Carefully check all the files related to print spooling (see “Null Modems for Printers and Plotters,” page 89), and then reboot the system.

6. In networks, you may have to change files on each remote host, unless the NIS service (previously called “yellow pages” service) is activated (see Chapter 17, “Networking via Ethernet,” for further information):

`/etc/hosts`	change remote hostname
`/etc/hosts.equiv`	change remote hostname
`/etc/vfstab`	(if used) change NFS lines to new hostname
`/etc/dfs/sharetab`	(if used, does not always contain host names)
`/.rhosts`	(if exists) change remote hostname

7. Remount all remote files (rebooting is not necessary):
   d. mountall -F nfs

8. If remote printing on the host with new name is used, carefully check and update all files related to printing (see “Null Modems for Printers and Plotters,” page 89).

2.7 Adding in Parts of Solaris “After the Fact”

Whenever possible, you should try installing everything you are going to need with the initial installation. This avoids complications such as disk partition overflow and more work; however, having a procedure for installing a package such as the UNIX manuals “after the fact” might save you from having to reinstall the entire software.

An example for such a situation is the following: because of disk space limitations, a user decides to only load the “End User Support” option. It turns out that this does not include the UNIX manuals. There are two ways to resolve this:

- After selecting “End User Support”, select the “Customize” button and then activate the “UNIX Manuals” selection before continuing with the software installation. This is the safest solution because it ensures that there will be enough disk space for the manual files in `/usr`, and it is definitely the simpler solution.
- Use the Sun `pkgadd` program to extract parts of the Solaris software from the CD-ROM “after the fact”, *provided there is enough disk space*.

Some precautions before we start: `pkgadd` does not report what the disk space requirements for the selected package are. You should only use this procedure if

- You know how much disk space is required for the package you want to load. Note that the UNIX manuals under Solaris 2.5.1 take up 14 MB of disk space (`/usr/man`). So you should make sure there is 14 MB of free disk space in `/usr`.
- You want to reload a specific package because the original files somehow got corrupted. In this case, no extra disk space should be required.

Assuming there is enough free disk space in `/usr`, we can start the installation:

1. Insert the Solaris 2.5.1 CD-ROM into the drive and continue as follows (you must be `root` to do this):

```
# cd /cdrom/cdrom0/s0/Solaris_2.5.1
# pkgadd -d .
The pkgadd program may require several minutes to start up.
After this, you see the first page filled with the first out of
about 200 software options:
The following packages are available:
  1  AXILvplr.c   Axil platform links
                  (sparc.sun4c) 5.5,REV=96.01.21.22.49
  2  AXILvplr.m   Axil platform links
                  (sparc.sun4m) 5.5,REV=96.01.21.22.50
  3  AXILvplu.c   Axil usr/platform links
                  (sparc.sun4c) 5.5,REV=96.01.21.22.51
  4  AXILvplu.m   Axil usr/platform links
                  (sparc.sun4m) 5.5,REV=96.01.21.22.51
  5  PFUaga.m     AG-10 device driver
                  (sparc.sun4m) 1.1.0,REV=0.0.2
  6  PFUagaow     AG-10 DDX Support(OpenWindows)
                  (sparc) 1.1.0,REV=0.0.2
  7  PFUagaoxgl   AG-10 Runtime Support
                  (sparc) 1.1.0,REV=0.0.2
  8  PFUcar.m     PFU/Fujitsu kernel/unix for Power Control
                  Software (sparc.sun4m)
                  11.5.1,REV=96.04.24.11.46
  9  PFUdbf.m     S-4/Leia LCD Dumb Frame Buffer Driver
                  (sparc.sun4m) 1.0,REV=96.04.18.01
 10  PFUvplr.m    PFU/Fujitsu platform links
                  (sparc.sun4m) 5.5,REV=96.01.21.22.42

... 182 more choices to follow;
<RETURN> for more choices, <CTRL-D> to stop display:
```

2. Press the Return key to scroll down until you find the online manuals:
   ```
    94  SUNWman      On-line Manual Pages
                     (sparc) 38.0,REV=7
   ```
3. At this point, stop the display by pressing Ctrl-d, which results in a dialog:
   ```
   Select package(s) you wish to process (or ‘all’ to process all
   packages). (default: all) [?,??,q]: 94
   ```
   This starts installation of the UNIX online manuals (generating lots of output, including copyright notes, etc.).
4. After the installation, `pkgadd` returns to the message in step 3. We stop the list with Ctrl-d and exit from `pkgadd`:
   ```
   Select package(s) you wish to process (or ‘all’ to process all
   packages). (default: all) [?,??,q]: q
   ```
5. We can now eject the CD-ROM—the installation is finished:
   ```
   # cd /
   # eject cdrom
   ```

Chapter 3. VNMR-Related System Administration

Sections in this chapter:

- 3.1 "Acquisition Processes," this page
- 3.2 "Stopping and Restarting Acquisition Processes," page 65
- 3.3 "Shutting Down and Restarting UNIX," page 68
- 3.4 "Configuring the VNMR User's Environment," page 71
- 3.5 "Shutting Down and Restarting the Host Computer," page 71
- 3.6 "Modifying the Nucleus Tables," page 72
- 3.7 "Changing the Standard Parameter Sets," page 73
- 3.8 "User-Selected Shim Buttons," page 73
- 3.9 "File Protection," page 74
- 3.10 "VNMR Accounting Tool," page 76

Administration of the spectrometer system and users on the system includes a number of activities that are covered in this chapter

3.1 Acquisition Processes

An understanding of the acquisition processes used by each system helps explain the need for the multiple-step start up and shut down procedures given later in this chapter. Acquisition is characterized by the interaction of a number of processes running on three interconnected computers:

- Sun-based host computer system.
- Ethernet controller on UNITY*INOVA*, *MERCURY* VxWorks Powered (shortened to *MERCURY-VX* throughout this chapter), *MERCURY*, and *GEMINI 2000*, or the Host-Acquisition Link (HAL) board on UNITY*plus*, UNITY, and VXR-S systems.
- Acquisition computer.

UNITY***INOVA* and *MERCURY-VX* Systems**

Figure 3 shows the relationship of acquisition processes on UNITY*INOVA* and *MERCURY-VX* systems. On the host computer system, the processes that may be involved are `acqi`, `Acqstat`, and the Proc family (`Expproc`, `Procproc`, `Roboproc`, `Autoproc`, `Infoproc`, `Recvproc`, and `Sendproc`). On the acquisition computer, the processes involved are VxWorks (the real-time operating system on the acquisition CPU), `vwacq.o` and `vwcomm.o`.

After the Sun computer boots, the acquisition computer uses `bootp` (boot protocol) over the Ethernet to the Sun to obtain the IP address and other information. The Sun finds the

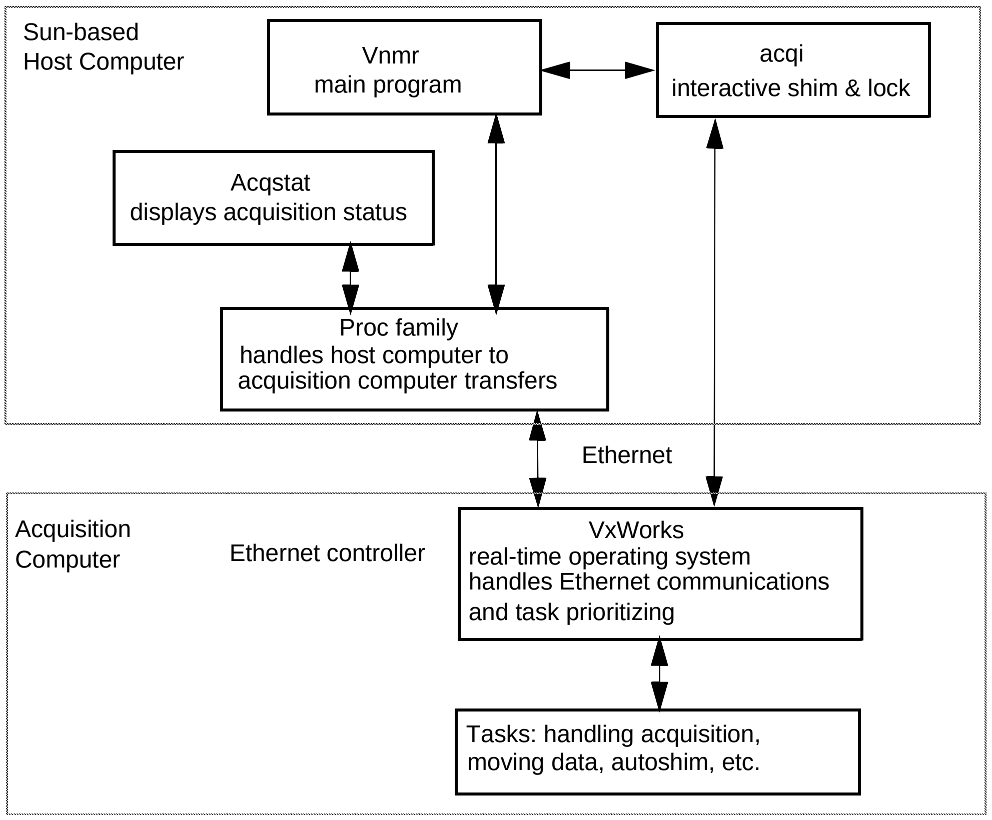


Figure 3. Acquisition Processes on UNITY*INOVA* and *MERCURY-VX*

information in the files `/etc/bootptab` and `/etc/hosts`, and returns the IP address and other information to the Ethernet controller.

The acquisition computer then requests VxWorks. The Sun computer obtains this from `/tftpboot/vxBoot` and sends it to the acquisition computer. After this, VxWorks requests programs for different acquisition tasks. The Sun gets these programs from the directory `/vnmr/acq`.

MERCURY and GEMINI 2000 Systems

Figure 4 shows the relationship of acquisition processes on *MERCURY* and *GEMINI 2000* systems. On the host computer, processes that may be involved are `acqi`, `Acqstat,` and `Acqproc`. On the acquisition system, processes involved are `lnc`, `apmon`, and `autshm`.

After the Sun computer boots, the acquisition computer is initialized for 1 second, and then waits for the Ethernet controller. The Ethernet controller initializes for 8 seconds and then uses RARP (Reverse Address Resolution Protocol) over the Ethernet to the Sun to obtain the IP (Internet Protocol) address. The Sun finds the address in the files `/etc/ethers` and `/etc/hosts`, and returns the IP address to the Ethernet controller.

The Ethernet controller then requests the `lnc` program. The Sun obtains this from `/tftpboot/lnc` and sends it to the Ethernet controller. Next, `lnc` boots the Ethernet controller. After this, `lnc` requests the `apmon` program and then the `autshm` program from the files `/tftpboot/apmon` and `/tftpboot/autshm`, respectively.

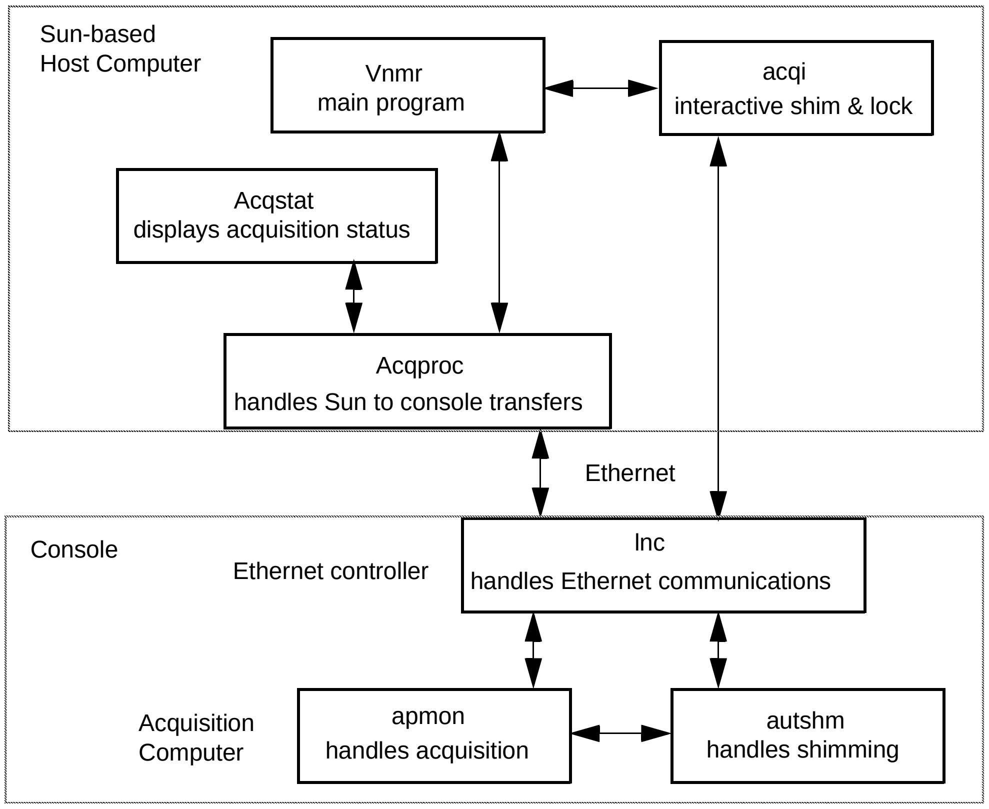


Figure 4. Acquisition Processes on *MERCURY* and *GEMINI 2000*

The Ethernet controller now passes information to the acquisition computer, which loads it in the correct place in the acquisition memory. Then the Ethernet controller sends a message to the acquisition computer to execute `apmon`. The `apmon` process initializes the hardware, finishing the system bootup. During operation, the `Acqproc` process polls the system every 5 seconds.

For more information on the acquisition processes used with the *MERCURY* and *GEMINI 2000*, refer to the manual *VNMR and Solaris Software Installation*.

UNITY*plus*, UNITY, and VXR-S Systems

Figure 5 shows the relationship between acquisition processes on UNITY*plus*, UNITY, and VXR-S systems. On the host system, the processes that may be involved are `acqi`, `Acqstat`, and `Acqproc`. On the acquisition system, the processes involved are `autshm` and `xrop` (VXR-S systems), `xrxrp` (VXR-S and UNITY systems), or `xrxrh` (UNITY*plus* systems).

When the systems are powered up, each computer goes through a startup sequence—the Sun starts up UNIX, the HAL starts up HALBUG and enters a "ready to receive" idle mode, and the acquisition computer starts up the PROM DIAGNOSTICS.

As one of the last startup operations, the Sun starts the `Acqproc` process, provided the file `/etc/acqpresent` exists (this is also true for *MERCURY* and *GEMINI 2000*). `Acqproc` attempts communication with the HAL through the SCSI bus and treats the HAL as a

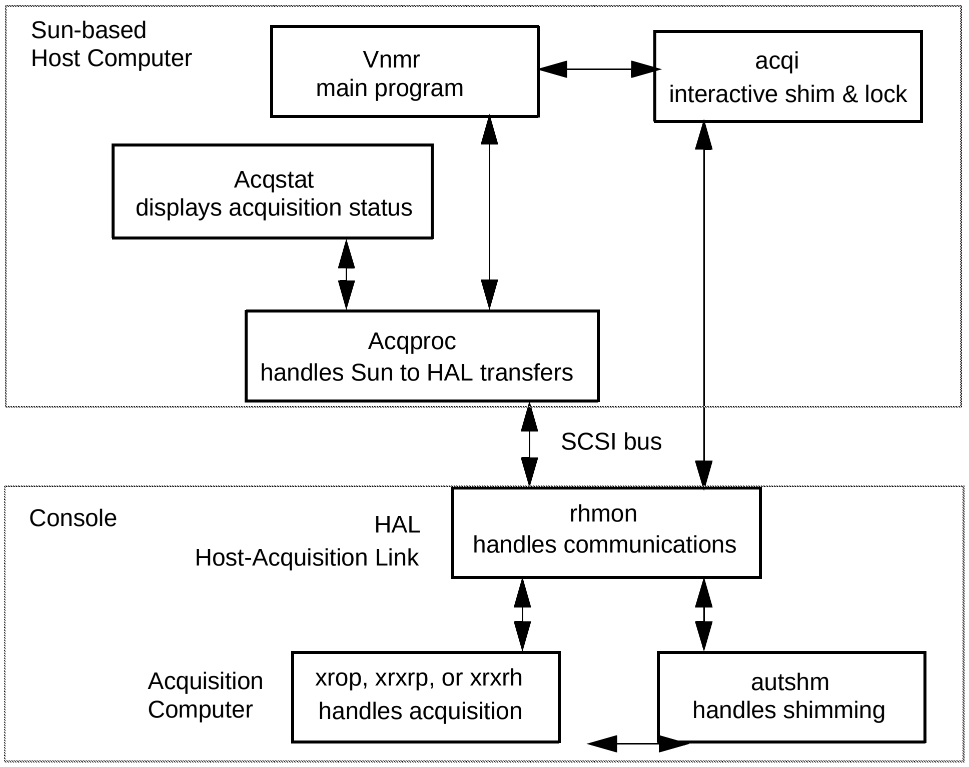


Figure 5. Acquisition Processes on UNITY*plus*, UNITY, and VXR-S

streaming tape device. Upon finding the HAL in the idle mode, `Acqproc` starts downloading of files from the `/vnmr/acq` directory.

The `Acqproc` process starts by downloading `rhmon.out` (the full HAL monitor) to HAL, followed by downloading `xr.conf` to the acquisition computer. `xr.conf` determines what hardware is present in the system and reports this back to the host computer. The information is saved in `/vnmr/acqqueue/acq.conf`, which is usable later by `config`.

Next `Acqproc` loads one of the following three files, as selected by `xr.conf`:

- `xrop.out` for VXR-S systems with a 63-word FIFO on the Output Board.
- `xrxrp.out` for VXR-S and UNITY systems with a 1024-word FIFO on the Acquisition Controller.
- `xrxrh.out` for UNITY*plus* systems. This module runs the acquisition process.

Finally, `Acqproc` downloads `autshm.out`, a software module used in automatic and interactive shimming. Once the download is complete, `Acqproc` drops into the idle mode, polling the HAL about every 5 seconds.

On all UNITY*plus*, UNITY, and VXR systems, a thumbwheel switch controls the output of diagnostic information from the acquisition computer to an optional terminal connected to the system. On UNITY*plus*, the switch is located on the Automation board and labeled *Boot Mode Select*. On UNITY and VXR-S systems, the switch is located on the Acquisition CPU board.

The switch positions, numbered from 0 to 9, have the following meaning (these settings also apply to the *MERCURY* and *GEMINI 2000* systems):

- 0 inhibits output to diagnostic screen except for errors. Time between transients is deterministic and significantly less than for positions 1 or 2.
- 1 prints diagnostic output using Televideo 920 cursor positioning commands. Time between transients is non-deterministic.
- 2 prints diagnostic output with no cursor positioning. Time between transients is non-deterministic.
- 3 boots the system directly to PROM debugger XRbug.
- 4 to 9 are undefined.

Except for a few systems that operate differently, position 0 is the normal setting. Setting the switch to other than 0 can increase significantly the time between FIDs and increments.

3.2 Stopping and Restarting Acquisition Processes

The standard operating configuration of VNMR software as a spectrometer controller runs a number of acquisition processes called the Proc family (`Infoproc`, `Procproc`, `Recvproc`, and `Sendproc` processes) on the UNITY*INOVA* and *MERCURY-VX*, or a process called `Acqproc` on other systems. Stopping and restarting these acquisition processes depends on the system:

- On the UNITY*INOVA* and *MERCURY-VX* systems, you can power down and restart the console at any time without executing special commands. In general, you can use on the UNITY*INOVA* and *MERCURY-VX* the same procedures to stop and start `Acqproc` as given in the following sections (including multiuser environments), but the commands involved will control the Proc family of processes rather than `Acqproc`.
- On the *MERCURY* and *GEMINI 2000* system, you can power down the console at any time without executing special commands. When turning off the *MERCURY* and *GEMINI 2000* console, `Acqproc` displays a warning message every 5 seconds on the Sun console. The messages stop when the console is powered back up.
- On UNITY*plus*, UNITY, and VXR-S systems, you must stop `Acqproc` according to the instructions in the following sections before powering down the acquisition system or the Host-Acquisition Link (HAL) board.

CAUTION: **On UNITY*plus*, UNITY, and VXR-S systems, if the acquisition computer or HAL board is to be shut down, Acqproc must be stopped first or else the system may hang up. Valuable data can be lost if a system reboot is then necessary to restart the system.**

Stopping Acqproc

Stopping the `Acqproc` process is required only on UNITY*plus*, UNITY, and VXR-S systems. It must be done before shutting down HAL, the acquisition computer, or the differential box (the unit on the SCSI cable between the magnet and the console on UNITY*plus*, UNITY, and VXR-S systems), or before shutting down the host computer so that a reboot is necessary. You must be logged in as `root`, the UNIX system administrator.

1. Log in as `root`, enter the `root` password (if implemented), and then enter:
```
# cd /vnmr/acqbin
# ./killacqproc
```

The system displays the messages:

```
Acqproc killed
```

2. After this message appears, it is safe to power down the HAL board (on UNITY*plus*, UNITY, and VXR-S systems only) or the acquisition computer.

   If the system does not return both of the messages shown above, repeat step 1. If a second attempt does not return the messages, take the following steps:

   a. Enter the `ps` command to show all active processes and (optionally) the `grep` command to show only the lines containing “`Acq`”:

      ```
      # ps -ef | grep Acq
      ```

   b. The system displays a table with five columns. Look at the entries in the `COMMAND` column (the last column) until you find an “`Acqproc`” entry. Then look at the number in the `PID` column (the first column) for the row containing “`Acqproc`.” Enter the following command, except replace `PID_number` with the number you found:

      ```
      # kill -9 PID_number
      ```

   c. Repeat the `ps` command you entered in step 2a and look again in the `COMMAND` column. “`Acqproc`” should be absent. If it still appears, repeat the `kill` command in step 2b. When “`Acqproc`” no longer appears, `Acqproc` has been stopped, and it is then safe to proceed.

3. One more step is required if the differential box is going to be powered down or if the host computer is going to be shut down so that a reboot will be necessary.

   When the `setacq` program is initially run on a system, it creates an empty file called `acqpresent` in `/etc`. On system bootup, the system looks for this file and, if it finds it, automatically starts `Acqproc`. If the HAL board has not been powered up and initialized, the system will hang. Therefore, `acqpresent` should be removed before shutting down the host computer or the differential box. While still logged in as `root`, enter:

   ```
   # cd /etc
   # mv acqpresent acqpresent.old
   ```

   This has the effect of renaming the file so that the bootup can not find it and does *not* start `Acqproc`. You can now safely shut down the host computer or the differential box.

Starting Acqproc

Perform step 1 only if the host computer or differential box was shut down after stopping `Acqproc`; if both are still powered on, perform step 2 only.

1. If the host computer or differential box were shut down after stopping `Acqproc`, boot up the host computer and power on the differential box. Then with UNIX running, log in as `root`, enter the `root` password (if implemented), and enter the following commands:

   ```
   # cd /etc
   # mv acqpresent.old acqpresent
   ```

2. Enter the `startacqproc` command as follows:

   ```
   # cd /vnmr/acqbin
   # ./startacqproc
   ```

Rebooting the Acquisition System

The acquisition system may occasionally need drastic action to recover. This normally occurs only during software development (e.g., executing a delay statement that inadvertently causes a division by zero) when the acquisition computer apparently "hangs" during a pulse sequence. Use the following procedure to reboot:

- On UNITY*INOVA*, press the reset button on the Acquisition CPU board. If the acquisition software (`Expproc` and Proc family) on the host computer is running, the shim DAC values (which are the last values returned before the console was rebooted) are automatically loaded into the console. After rebooting the acquisition system, you must enter `su` before using `Acqi` or `qtune`.
- On *MERCURY*-Series, press the reset switch behind the console door.
- On *GEMINI 2000*, lift the reset switch behind the console door.
- On UNITY*plus*, UNITY, and VXR-S, the VNMR command `abortallacqs` reboots the acquisition computer and downloads again the files `xr` and `autshm`, which reset the acquisition system after any catastrophe. However, HAL must be operational for `abortallacqs` to work. If this is not the case, `Acqproc` issues a message to the effect "message not sent, retry later." If the acquisition system is rebooted while `Acqproc` is running, `Acqproc` senses this and, when the reboot is complete, downloads the files as described in "Acquisition Processes," page 61.

Controlling the Acquisition Process in Multiuser Environments

Typically, `Acqproc` (or `Expproc` and Proc family on UNITY*INOVA* and *MERCURY-VX*) is started by `root`, which means that any user who needs to kill or start the acquisition process must know the `root` password. In a multiuser environments, this is not viable.

An alternate method of stopping and starting the acquisition process is to install and use the special login name `acqproc`. This command, available on all systems including UNITY*INOVA* and *MERCURY-VX*) starts a script, called `execkillacqproc`, that can start or kill the acquisition process, as follows:

- If the acquisition process is not running, `acqproc` starts it (with `root` privileges).
- If acquisition process is running, `acqproc` stops it.

To install the `acqproc` login name on a system, take the following steps (it is assumed that `makevnmr1` has been already run during the installation):

1. Log in as `root`, enter the `root` password (if implemented), and enter the following commands:
   ```
   # cd /vnmr/bin
   # ./makesuacqproc
   ```
   The `makesuacqproc` command installs the login name `acqproc`.

   If necessary, the system displays a prompt requesting the following information:
   ```
   Please enter location of VNMR system directory
   (default: /vnmr)
   ```
2. Press the Return key to accept the default directory.

   The system prompts:
   ```
   acqproc is now a login that will start or kill Acqproc.
   ```
3. The acquisition process can now be started or killed from the login prompt
   ```
   login: acqproc
   ```
   or by any user
   ```
   varian> su acqproc
   ```

3.3 Shutting Down and Restarting UNIX

The following sections illustrate the safest methods for shutting down UNIX in an orderly fashion so as to minimize the risk of corrupting hard disk data. It is not advisable to shut down UNIX unless absolutely necessary, for example, system maintenance or repair, planned power removal, or moving the system.

CAUTION: **If proper shutdown procedures are not followed, system and user files stored on hard disks can be corrupted.**

Normal UNIX Shut Down

To shut down the system, you must be logged in as `root` from the `login`: prompt or you must have entered `su` command (not from a shell).

If you are currently in VNMR, exit VNMR first before starting here.

In the procedure below, entering the `killacqproc` command is not needed if you are only rebooting the system; the reboot will kill `Acqproc`. Also, step 2 to rename the `acqpresent` file is only needed if `Acqproc` is not wanted when rebooting; for instance, when the NMR console will not be used.

1. Log in as `root`, enter the `root` password (if implemented), then enter the following commands to stop the `Acqproc` process:
   ```
   # cd /vnmr/acqbin
   # ./killacqproc
   ```
   The system communicates success by displaying:
   ```
   Acqproc killed
   Acquisition daemon terminated
   ```
   If both of these messages appear, it is safe to go to step 2.

   If *both* of the messages do not appear, repeat step 1. If both of the messages do not appear after a second attempt to stop the `Acqproc` process, take the following steps:

   a. Enter the `ps` command to show all active processes and (optionally) the `grep` command to show only the lines containing “`Acq`”:
      ```
      # ps -ef |grep Acq
      ```
      The system displays a table with five columns. Look at the entries in the `COMMAND` column (the last column) until you find an “`Acqproc`” entry. Then look at the number in the `PID` column (the first column) for the row containing “`Acqproc`.”

   b. On systems other than UNITY*INOVA*, enter the following command, except replace `PID_number` with the number you found:
      ```
      # kill -9 PID_number
      ```
      For UNITY*INOVA*, it is `Expproc`, not `Acqproc`, that must be killed. Instead of the last command enter:
      ```
      # kill –2 PID_number
      ```
      When `Expproc` is killed this way, by using `–2`, the complete Proc family (`Expproc`, `Recvproc`, etc.) is stopped.

   c. Repeat the `ps` command you entered in step 1a and look again in the `COMMAND` column. “`Acqproc`” should be absent. If it still appears, repeat the `kill` command in step 1b. When “`Acqproc`” no longer appears, the `Acqproc` process has been stopped, and it is then safe to proceed.

2. Rename the file `acqpresent` as follows:
   ```
   # cd /etc
   # mv acqpresent acqpresent.old
   ```
3. If you want to shut down immediately, enter:
   ```
   # /etc/shutdown -y -g0 -i0
   ```
   If you want to shut down at the given time instead of immediately, set the number of seconds before shutdown using `/etc/shutdown –y –gsec –i0`, where `sec` is the number of seconds. For example, for 30 minutes to shutdown, enter the command `/etc/shutdown –y –g300 -i0`. For more information on `shutdown`, see Solaris documentation or enter `man shutdown`.

   At shutdown, the system forces any information on its way to the hard disk to be written out immediately, cleans up any processes that are running, and executes an orderly shutdown of the system. The process takes about 20 seconds. When the system has safely shut down, the prompt "`>`" or "`OK`" appears.
4. It is now safe to turn off the system power. Turn off the power switches in this order:
   a. Hard disk(s) and tape drive(s) (if drives have separate power switches).
   b. CPU unit.
   c. Monitor.
5. If your system is a SPARCstation 1+ or 1, you *must* turn off the differential box at this time. If your system is a system other than a SPARCstation 1+ or 1, turn off the differential box at this time only if necessary. (The UNITY*INOVA*, *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* do not have a differential box.)

Shutting Down UNIX in an Emergency

If no one can log on as `root` (e.g., the password is lost), and you absolutely must reboot or remove power from the UNIX system, it is necessary to run the risk of corrupting files and possibly losing data. The following procedure is not safe, but in an emergency it is marginally better than simply turning the power off.

1. Exit VNMR, and then exit the window manager, if possible.
2. Synchronize the disks by entering:
   ```
   > sync;sync;sync
   ```
3. Press Stop-a (hold down the Stop key and press the a key). On older keyboards, the Stop key is labeled L1, so press L1-a instead.

The `sync;sync;sync` command attempts to ensure that all disk writes are completed but does not do it as thoroughly as the `shutdown` command. The L1-a key combination is a system reset that should be used only in an emergency when nothing else works. Files can be lost using the L1-a reset.

Restarting UNIX

Start with step 1 below if restarting (rebooting) UNIX after power has been removed from one or more system components. If no units are powered down, you can skip steps 1 and 2, and restart UNIX from the boot prompt "`>`" by typing the command `b` and following the instructions in steps 3 and 4 below. From the "`ok`" prompt, type `boot`.

You do not need to perform step 4 in this procedure if you did *not* rename the `acqpresent` file when shutting down.

1. After the ac power cables have been reconnected, turn on the power switches to each component of the system in this order (the UNITY*INOVA*, *MERCURY-VX*, *MERCURY*, and *GEMINI 2000* do not have a differential box):
   - For all Sun systems except the SPARC station 1+ and 1:

     Differential box (if it was necessary to turn off during the shutdown)
     Monitor
     CPU unit
     Hard disk(s) and tape drive(s) (if the drives have separate power switches)
   - For the SPARCstation 1+ and 1 (*do not turn on the differential box yet*):

     CPU unit
     Monitor
     Hard disk(s) and tape drive
2. Upon turning on the CPU unit, the following happens:
   - For all Sun systems except the SPARC station 1+ and 1:

     System performs a standard self-test, then executes an automatic boot procedure
   - For the SPARCstation 1+ and 1:

     The system performs a standard self-test, then displays the following:
     ```
     Testing
     Type b(boot), c(continue), or n (new command mode)
     ```
     When this message appears, turn on the power to the differential box, and type `b` to boot the system (if the prompt "OK" appears, type `boot` instead):
     ```
     > b
     ```
3. When the system has successfully booted, the `login` prompt or CDE login screen appears. Log in as `root` and enter the `root` password (if implemented).

   If you renamed the `acqpresent` file when the system was shut down, enter the following commands to rename `acqpresent` and execute `startacqproc`:
   ```
   # cd /etc
   # mv acqpresent.old acqpresent
   # cd /vnmr/acqbin
   ```
4. Execute `startacqproc`:
   ```
   # ./startacqproc
   ```
5. If you want to log in as another user (e.g., user `vnmr1`, the NMR system administrator), enter the `logout` command, then log in using another valid user name and enter the user password (if implemented):
   ```
   # logout
   login: username
   Password: user_password
   ```

Note: *The following applies to the SPARCstation 1+ or 1 only:*

When powering on the SPARCstation 1+ or 1, always start with the differential box power off until the following message appears:
```
Testing
Type b(boot), c(continue), or n (new command mode)
```
If the differential box and the acquisition console both have their power on when the SPARCstation 1+ or 1 is powered on, the Sun computer may not boot, showing this message:
```
Extra scsi data. Fatal error.
```
Any command entered at this point results in the message:

`Data Access Exception`
You must turn off the SPARCstation 1+ or 1 as well as the differential box and reboot the acquisition console before proceeding.

3.4 Configuring the VNMR User's Environment

The standard VNMR installation toolkit for UNIX contains the `makeuser` command, which creates a standard environment for working with VNMR.

Adding a User

This `makeuser` command adds an account (user name) to the UNIX `passwd` system and creates a home directory for that account. `makeuser` also places the files and subdirectories necessary to run VNMR into the user's home directory. In addition to the standard VNMR files and directories, the following UNIX "dot" files (files whose name starts with a period) are put into the HOME directory:

- On all systems: `.cshrc` and `.login`.
- On X Window versions: `.openwin-menu`, `.openwin-init`, and `.Xdefaults`.
- On the CDE user interface: `.dtprofile` and `.dtdirectory`.

The standard UNIX `ls` command to list files does not include dot files in the display—you must use the `-a` option to `ls` (i.e., `ls -a`) to display them.

The `makeuser` command is also used to update user accounts when new software versions are installed. For details, consult the installation manual for the update.

Resizing Windows

The major windows in VNMR automatically resize themselves based on the size of the text characters used. The menu bars, graphic window, text output, and status windows automatically size themselves to be 80 characters wide, while the interactive acquisition windows are 32 characters wide.

On X Window versions, the font used for display of text on the screen is contained in the file `.Xdefaults`, which is in `/vnmr/app-defaults`. `Xdefaults` can be edited with any UNIX text editor. One of the lines in the file (`*VNMR*font`) sets the size of the font. A good size for the font to make all windows fit (not overlap) is 9x15, so we recommend editing `.Xdefaults` to have the line

```
*VNMR*font       9x15
```

3.5 Shutting Down and Restarting the Host Computer

If the host computer must be removed from the main network, put on a main network or have its network parameters changed, this section describes the procedure for performing any of those operations.

You must use the command `sys-unconfig` when rebooting the host computer. *Before* issuing this command, refer to the manual *VNMR and Solaris Software Installation* for *all* information pertaining to the network, including server name, server Internet protocol (IP) address, and netmask, and any other necessary information.

1. Completely log out of the system.

2. Open a UNIX shell window (or command tool) and log in as `root`.
3. Enter the `sys-unconfig` command:
   ```
   # sys-unconfig
   ```
4. You are prompted to confirm that you want to shut down the system. If you confirm that you do, the system shuts down.
5. Reboot the system by entering `boot` at the `ok` prompt:
   ```
   ok boot
   ```
   A series of questions on screens similar to the screens displayed when Solaris was installed appear. You must answer the questions to be able to boot the system.
6. When prompted, enter a new root password.
7. After the system has been rebooted, execute `setacq` as described in the manual *VNMR and Solaris Software Installation*.

3.6 Modifying the Nucleus Tables

Varian NMR spectrometers use the concept of a nucleus table, in which the parameters `tn` and `dn` are looked up and translated into a transmitter or decoupler frequency. The nucleus tables for the different types of rf and the different field strengths are organized into a directory named `/vnmr/nuctables`. The actual nucleus table used is derived from the `conpar` parameters `h1freq` and `rftype` of the instrument. For example, a 300-MHz system with `rftype` set to `c` for transmitter and set to `b` for decoupler would use `nuctab3c` and `nuctab3b`, respectively. Correctly configuring the rf type using the `config` program results in the correct nucleus tables being used on your system.

If it appears that the proper frequencies are not being used on your spectrometer, start by displaying the nucleus table (e.g., `cat('/vnmr/nuctables/nuctab3c')`). If the frequencies are proper, the problem is probably elsewhere (check the value of `lockfreq` among other things). If the frequencies are not proper, use `config` to check the configuration parameters.

Table 7 shows the format of nucleus table text files for a channel.

Table 7. Format of a Nucleus Table

Name	*Frequency*	*Baseoffset*	*FreqBand*	*Syn*	*RefFreq*
`H1`	`599.9450000`	`1.4803000e+06`	`high`	`yes`	`0.0`
`H2`	`92.0930000`	`1.4803000e+06`	`low`	`yes`	`0.0`
`H3`	`639.9286000`	`1.4803000e+06`	`high`	`yes`	`0.0`
`He3`	`457.0098000`	`1.4803000e+06`	`high`	`yes`	`0.0`
`...`	`...`	`...`	`...`	`...`	`...`

The columns in these files have the following meaning:

- `Name` lists the nucleus name, which is entered as `tn` or `dn`.
- `Frequency` is the spectrometer frequency, in MHz.
- `Baseoffset` is no longer used and can be set to any value.
- `FreqBand` is the frequency amplifier band for the channel: `high` or `low`.
- `Syn` is `yes`.

- `RefFreq` is reserved for future use.

The nucleus files may be modified by the NMR system administrator using any of the available text editors. The order of the columns must be kept, but spacing between entries can be altered. It is recommended that trailing zeroes are kept because this indicates the true precision of the entry of 0.1 Hz.

3.7 Changing the Standard Parameter Sets

The directory `/vnmr/stdpar` contains a series of files, each named after a nucleus, that contain the "standard" parameter sets for that nucleus. These files have names like `stdpar/H1`, `stdpar/C13`, and so forth. To display a complete list, enter the command `ls('/vnmr/stdpar')`. These parameter sets are used by the command `setup`, as well as by many of the existing macros, to set initial parameters for that particular nucleus.

When you initialize your system and periodically later, you may need to change these parameters for the following reasons:

- You want to choose different default parameters. For example, the `stdpar/H1` parameter set calls for a 3000 Hz spectral width and you prefer 4000 Hz as a default.
- You want to update the parameter sets so that they reflect current calibration.

Change parameters in `stdpar` as follows:

1. Use `rtp` to recall the parameter set (e.g., `rtp('/vnmr/stdpar/H1')`).
2. Change the parameters of interest (e.g., `vs=14.3`).
3. Use `svp` to save the changes (e.g., `svp('/vnmr/stdpar/H1')`).
4. The system displays the message:
   ```
   File exists, overwrite (enter y or n <return>)?
   ```
   Since you do wish to overwrite the existing file, type `y` and the parameter set is updated; otherwise, type `n`.

The system standard parameter sets can be modified by the `vnmr1` user only.

Each user can also have personalized standard parameters in a personal directory (e.g., `/export/home/vnmr1/vnmrsys/stdpar`). If such files exist, they take precedence over the system files in `/vnmr/stdpar`, and hence changes made in `/vnmr/stdpar` only affect those users lacking a personal file.

Parameters `dmf`, `dpwr`, `pp`, `pplvl`, `pw90`, and `tpwr` are now taken from a probe calibration file.

3.8 User-Selected Shim Buttons

For setting up menus and shim groups within `acqi`, a definition file is searched for, first in the user's private directory and then in the `acq` subdirectory of the VNMR system directory. The name of the definition file is `acqigroup1`, `acqigroup2`, `acqigroup3`, `acqigroup4`, `acqigroup5`, `acqigroup6`, or `acqigroup7`, according to whether the parameter `shimset` is 1, 2, 3, 4, 5, 6, or 7, respectively (8, 9, and 10 are done differently). `shimset` is a configuration parameter used to set the shim set mode in the `config` program.

The definition files are text files that contain instructions on how to modify the `acqi` menu and shim groups. Note that the `acqi` program only reads the definition file once, when the program first starts. If you want to customize the file, do it before `acqi` is started or else after the file is customized, exit and restart `acqi`.

In the interactive SHIM and FID display, the shim groups and menus in the two bottom panels are user-selectable. On the interactive shim buttons panel, the Manual Shim menu can be user programmed and the shims can be grouped by the user. The same panel is used for the SHIM and FID display. On the Autoshim panel, the Autoshim menu can be user selected. This is specified in the same file.

3.9 File Protection

VNMR 5.3 and later interacts with the `umask` file protection system of UNIX. The major advantage of this feature is that it allows the VNMR system administrator to configure a spectrometer or data station so that multiple accounts can share the same VNMR user directory.

Each time a user logs in to UNIX, the system assigns a file-creation permissions mask (a `umask)` for that login. This mask sets those bits that are to be *always* cleared in the file protection field whenever a file is created. File protection determines what level of access is available to a file. Access can be read, write, or execute. A bit is assigned for each type of access. Therefore, three bits are required to specify access to a file.

If the bit is set, the corresponding access is permitted; access is denied if the corresponding bit is clear. Because three bits are equivalent to one octal digit, file access on UNIX is specified as an octal digit. At the basic level, the digits are assigned as follows:

- Octal digit 1 is execute permission.
- Octal digit 2 is write permission.
- Octal digit 4 is read permission.

By combining these values, levels of access can be established using a single octal digit:

- A value of 7 (4 + 2 + 1) means any kind of access is permitted—read, write, or execute.
- A value of 0 means no access at all is permitted.
- A value of 5 (4 + 1) means read and execute access.
- A value of 6 (4 + 2) means read and write but no execute access. This value applies to data files that are not to be run as programs.

Access is further determined based on the owner of the file and the group to which the file belongs. (Use the UNIX command `ls -lg` to list the group associated with a file.) Different access levels may be set for the file owner, other members of the same group, and any user on the system, respectively referred to as *owner*, *group*, and *world* or *everyone*.

With three different levels of access, three octal digits, such as 000 or 022, are necessary to specify file protection. Both file protection itself and `umask` are expressed with the first digit for the owner, the second digit for the group, and the last digit for the world. In C programming language, a numeric value preceded by a 0 represents an octal value. Note that the `umask` command does not require this preceding 0, and when it displays the current `umask` it omits this digit.

The default VNMR `umask` value is 022. Recall that the bits set in the `umask` are the ones that will be cleared in the file protection mask. In this case, write access to the group and the world is automatically denied. With a `umask` of 000, everyone on the system can access everyone else's files. No bits are cleared in the file protection and no access is denied by default.

If all members of a group are to share access to the same VNMR user directory, set `umask` to 002. If everyone who accesses the system shares the same VNMR directory, set `umask` to 000. set `umask` in the `.cshrc` file (`.profile` if the Bourne shell is the default shell), *not* in the `.login` file. Provide the desired value as an argument to the `umask` command.

Hiding a Command

The VNMR command `hidecommand` can rename built-in commands, permanently remove user access to commands, temporarily disable commands, and list buttons.

- `The hidecommand` command renames (or hides) a built-in VNMR command so that a macro with the same name as the built-in command is executed instead of the built-in command. As an example, suppose a user wrote a macro named `svs` to saves shims to a predefined directory. In addition to the `svs` macro, there is also a built-in `svs` command that saves shims in the `shims` directory. The user enters `svs`. Because built-in commands are always executed first, the `svs` command is executed instead of the `svs` macro, and the shims are not saved in the user's predefined directory. The user can easily solve this problem by entering `hidecommand('svs')`before entering `svs`.

  The built-in `svs` command is renamed (`svs` is automatically changed to `Svs`). The next time the user enters `svs`, the macro is executed before the `svs` command. The new name of the command can be obtained by entering `hidecommand('svs'):`*`$new_name`*`.`

  To reset a built-in command to its original name, enter `hidecommand` with the hidden name, for example, `hidecommand('Sys')` changes `Svs` to `svs`.
- Administrators can use `hidecommand` to remove access to commands. For example, an administrator could deny users access to the `rm` command by including the line `hidecommand ('rm')` in the system bootup macro.
- `The hidecommand` command can be used to temporarily disable commands. In the following script, the first command hides the built-in `svs` command. The second command executes the user's `svs` macro. The third command executes the built-in `svs` command with its new name. The fourth command resets the `svs` command to its original name. The fifth command executes the built-in `svs` command:

  ```
  hidecommand('svs'):$new_name
  svs
  exec($new_name)
  hidecommand($new_name)
  svs
  ```
- Entering `hidecommand` with a question mark as an argument displays a list of all the renamed button commands.

3.10 VNMR Accounting Tool

The VNMR Accounting tool provides NMR administrators with an easy way to keep a log of VNMR users. The program gives you the following capabilities:

- Create groups of console users with single-rate or multi-rate billing.
- Calculate special rates on days identified as holidays on a calendar.
- Show and print lists and accounting reports.

This section includes the following information.

- "Tutorial: Using the Accounting Tool," next
- "Navigating the VNMR Accounting Tool," page 84

Tutorial: Using the Accounting Tool

As an example, let's look at how Dan sets up an accounting group for billing purposes.

Dan has six users:

- Himself (dans)
- nmrguy
- root
- spinguy
- vnmr1
- writer1

He decides to set up five groups:

- Standard (dans, writer1, single rate)
- Special (spinguy, multiple rate)
- nmrguy (single user)
- Maintenance (no users)
- Administration (root, vnmr1).

To accomplish this task, Dan does the following procedures:

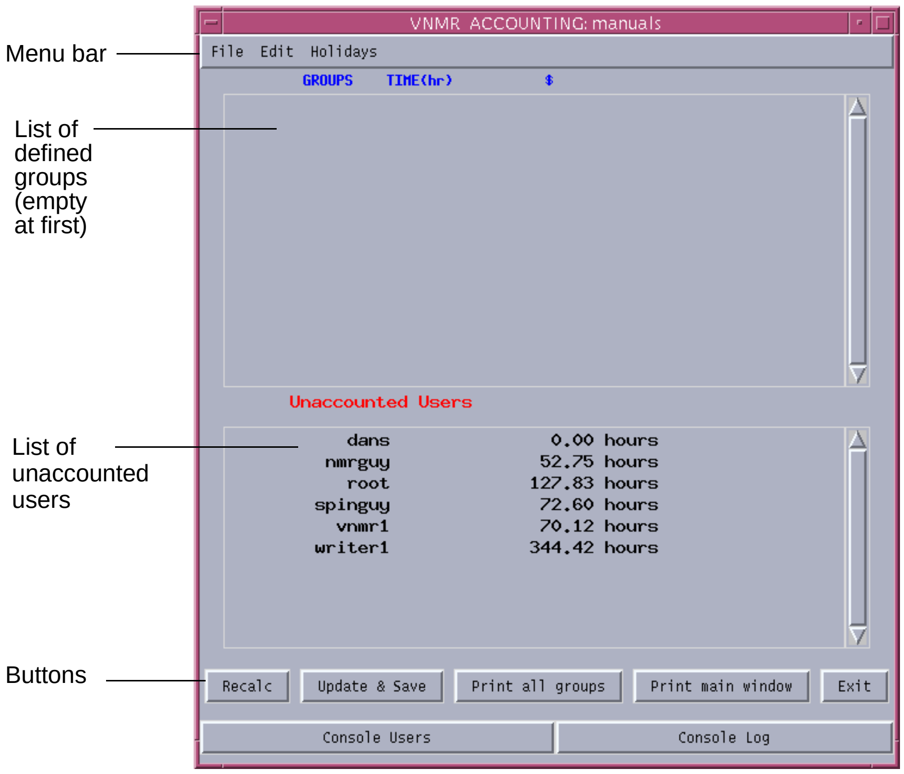
- "Opening the Accounting Tool," next
- "Setting Up Single-rate Users," page 77
- "Setting Up Multiple-rate Groups," page 78
- "Setting Up the Weekday Rate," page 79
- "Setting Up the Weekend Rate," page 79
- "Setting Up the Holiday Rate," page 80
- "Adding Users to Groups," page 80
- "Calculating Billing Information," page 81
- "Seeing a List of Users and Groups," page 81
- "Seeing a Chronological User Log," page 82
- "Seeing and Printing Reports," page 82

Opening the Accounting Tool

To open the accounting tool, Dan can perform one of two actions:

- In the main menu of theVnmrJ Admin tool, he clicks on **Management**, then **Cost/ Time Accounting**.

  A window similar to Figure 6 opens.



```
   dans        0.00 hours
 nmrguy       52.75 hours
   root      127.83 hours
spinguy       72.60 hours
  vnmr1       70.12 hours
writer1      344.42 hours
```

Figure 6. VNMR Accounting Window

- He can open the accounting program without running VNMR. In a terminal or UNIX shell, he can log in as **`root`**, then enter the command **`/vnmr/bin/vnmr_accounting`**

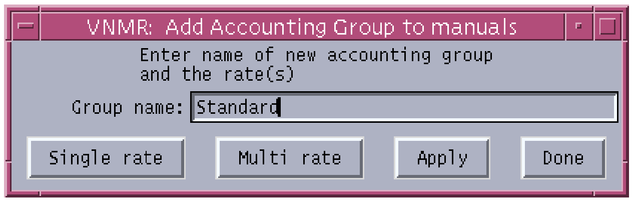
Note: Users other than `root` can open the program for viewing but cannot make changes to the data.

Setting Up Single-rate Users

To set up single-rate groups, Dan does the following steps.

1. From the **Edit** menu, he selects **Add_Group**.

2. In the window shown in Figure 7, he types **`Standard`** as the name of the new accounting group in the **Group name** field.



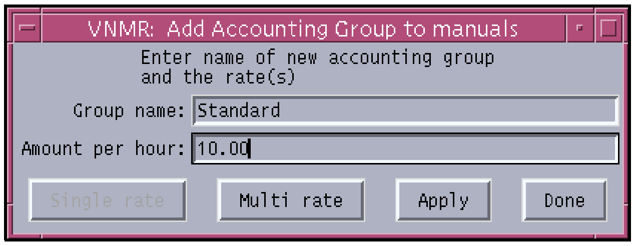
Enter name of new accounting group
and the rate(s)

Group name: Standard

Figure 7. Add Accounting Group Window

3. Because the Standard group bills at a single rate, he clicks on the **Single rate** button

   A window opens similar to Figure 8.



Enter name of new accounting group
and the rate(s)

Group name: Standard

Amount per hour: 10.00

Figure 8. Add Group Window, Single Rate

4. In the **Amount per hour** field, he types **`10.00`** for the billing rate, and then clicks the **Apply** button. The upper pane now lists the Standard group.
5. Dan repeats steps 2 and 3 to add the nmrguy, Maintenance, and Administrative groups, which are all single-rate groups, and set rates. Each is listed in the upper pane.
6. When he is finished adding groups, Dan clicks the **Done** button.

Tip: To simplify accounting for the `root` user, Dan sets up a group named “Maintenance” with 0 as the charged rate. Then, he assigns `root` to this group.

Setting Up Multiple-rate Groups

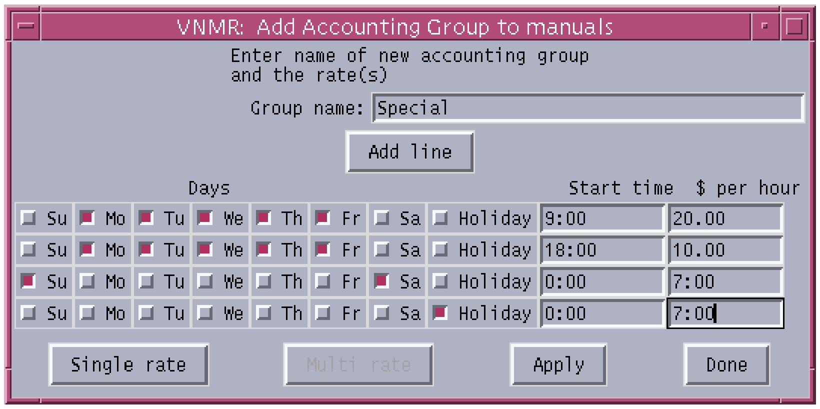
Now Dan wants to set up a multiple-rate group.

1. From the **Edit** menu, he selects **Add_Group**.
2. In the **Group name** field, he types **`Special`** as the name of the new accounting group.
3. He clicks the **Multi rate** button.

   A window opens similar to Figure 9.

   Dan wants to set the following rates for the Special group:

   - From 9:00 a.m. to 5:59 p.m. on weekdays, time is billed at $20.00 per hour.
   - Weekday evening time is billed at $10.00 per hour.
   - Weekends (Saturday and Sunday) are billed at $7.00 per hour.
   - Holidays are also billed at $7.00 per hour.


VNMR: Add Accounting Group to manuals

Enter name of new accounting group
and the rate(s)

Group name: Special

Days								Start time	$ per hour
Su	[✓ Mo]	[✓ Tu]	[✓ We]	[✓ Th]	[✓ Fr]	Sa	Holiday	9:00	20.00
Su	[✓ Mo]	[✓ Tu]	[✓ We]	[✓ Th]	[✓ Fr]	Sa	Holiday	18:00	10.00
[✓ Su]	Mo	Tu	We	Th	Fr	[✓ Sa]	Holiday	0:00	7:00
Su	Mo	Tu	We	Th	Fr	Sa	[✓ Holiday]	0:00	7:00

Figure 9. Add Group Window, Multi Rate

Setting Up the Weekday Rate

1. To set up the weekday rate, Dan clicks the **Add line** button once, and then clicks on **Mo**, **Tu**, **We**, **Th**, and **Fr** in both lines.
2. In the first line, he types **9:00** in the **Start time** field and **20.00** in the **$ per hour** field. In the second line, he respectively types **18:00** and **10.00**.

   Dan can add as many rates as he wants by creating rows. To create rows, he clicks the **Add line** button.
3. When he has completed entering rates, Dan clicks **Done**.

Setting Up the Weekend Rate

1. To set up the weekend rate, Dan clicks the **Add line** button once, and then click on **Su** and **Sa**.
2. He types **0:00** in the **Start time** field and **7.00** in the **$ per hour** field.

   Note that the billing rate begins at the start time and runs until the next start time.Therefore, billing for Sunday night to Monday morning is at $7.00 per hour until 8:59 a.m. on Monday. Monday night billing (starting at 6:00 p.m. at $10.00 per hour) runs through to 8:59 a.m. Tuesday. It is not necessary to start an accounting day at 0:00; the preceding day continues until the first change.

Setting Up the Holiday Rate

1. To set up the holiday rate, Dan clicks the **Add line** button once, and then clicks on **Holiday**.
2. He types **0:00** in the **Start time** field and **7:00** in the **$ per hour** field.
3. Finally, he clicks on the **Apply** button.
4. Dan could add other multiple-rate groups now, but because he has only one group, he clicks on the **Done** button.

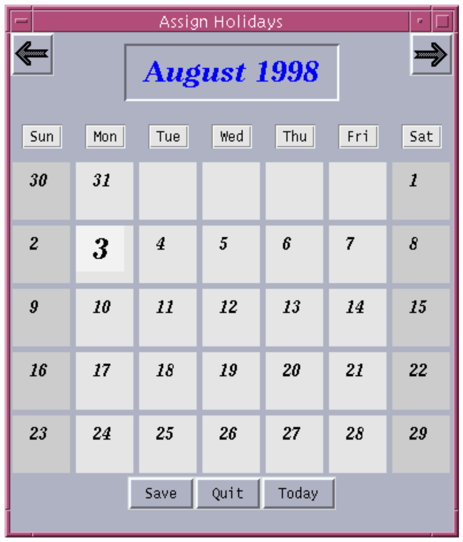


Figure 10. Assign Holidays Window

Designating Holidays

Because the multiple-rate group has a billing rate for holidays, Dan needs to designate the days that are holidays.

1. From the **Holidays** menu, he selects **Add**. A monthly calendar similar to Figure 10 appears:

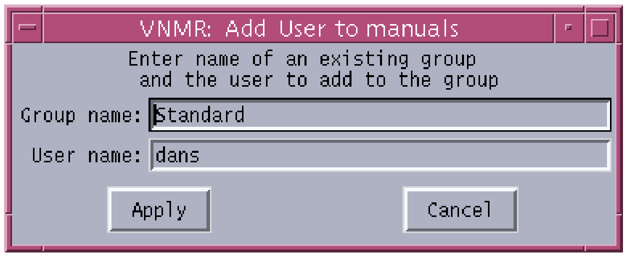
   Today's date appears as a large, bold number. The arrows at the top of the window move the calendar forward and backward. To return to today's date, Dan clicks the **Today** button.
2. With the left mouse button, he clicks on each day that he wants to assign as a holiday. The dates change to red to indicate a holiday. There is no limit to the number of days that Dan can assign as a holiday.
3. When he is done, he clicks the **Save** button and then the **Quit** button.

Adding Users to Groups

Dan can add users as either individuals or as members of a group. An easy way to define users as individuals is to create a group with the same name as the user and only assign that single user to the group (e.g., Bob Smith is the only member in the group Bob Smith). After Dan adds a group, its name appears in the upper pane.

1. To add users to the group, Dan selects **Edit** in the main menu, the **Add_User**.

   A window, similar to Figure 11, opens.
2. In the **Group name** field, he types **`Standard`**.
3. He wants to add dans and writer1. In the **User name** field, he types **dans**, then clicks the **Apply** button.
4. He repeats step 3 to add writer1.



VNMR: Add User to manuals

Enter name of an existing group
and the user to add to the group

Group name: Standard

User name: dans

Figure 11. Add User Window

Dan repeats steps 2 and 3 for each group until each user is in a group.

As Dan assigns users to a group, their names are removed from the Unaccounted Users list. If you assign all users, no names should appear in the Unaccounted Users pane. Assigning all users is recommended because it makes it easy to identify any unauthorized spectrometer access, because unauthorized users will appear in the Unaccounted User pane.

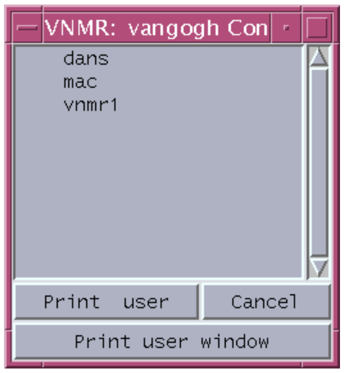


Figure 12. Console Users Window

Adding Users to an Existing Group

To add one or more users to an existing group, Dan does the following steps.

1. From the **Edit** menu, he selects **Add_User** to open a window similar to Figure 11.
2. In the **Group name** field, he types the name of the accounting group.
3. In the **User name** field, he types the name of a user to add to the group. To enter multiple users, he separates each name with a space.
4. To save his changes he clicks the **Apply** button.

Calculating Billing Information

Now that the groups and users have been set up, Dan wants to calculate the billing information. He does so by clicking the **Recalc** button.

Progress is displayed on the main screen. If there are many entries to be processed, this process can be time-consuming.

Accounts are not automatically updated so Dan needs to click on **Recalc** whenever he needs to update billing information.

Seeing a List of Users and Groups

To see a list of users, Dan clicks the **Console Users** button. A window similar to Figure 12 appears.

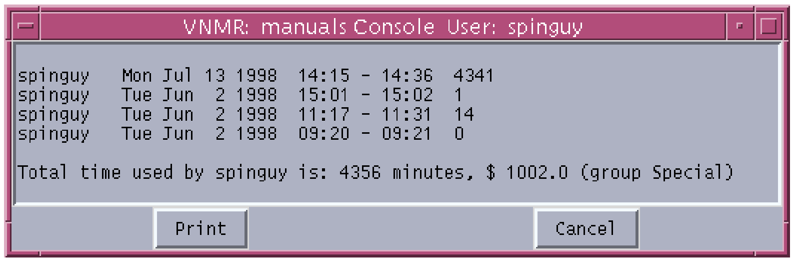
The list of established groups is displayed in the top pane of the tool. To see more information about any group listed, Dan double-clicks the name of the group.

Seeing Detailed User Information

To see a list of console users and user information, Dan does the following steps:

1. He clicks the **Console Users** button to open a window similar to Figure 12.
2. He double-clicks the name of the user to open a window similar to Figure 13.

   User information includes login names, the day, month, and year that users were logged in; login and logout times, in hours and minutes; and a total of the time users were logged on to the console. This information is the most recent data since the last calculation of time and expenditure summaries for the known groups.



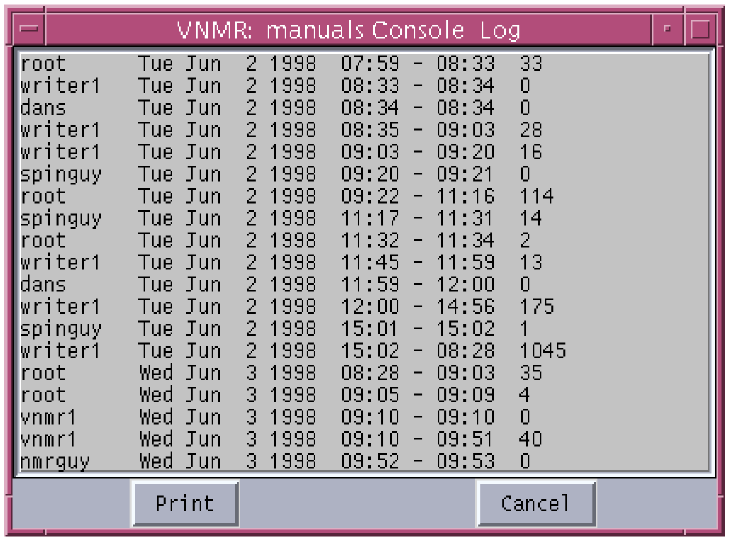
```
spinguy   Mon Jul 13 1998  14:15 - 14:36  4341
spinguy   Tue Jun  2 1998  15:01 - 15:02  1
spinguy   Tue Jun  2 1998  11:17 - 11:31  14
spinguy   Tue Jun  2 1998  09:20 - 09:21  0

Total time used by spinguy is: 4356 minutes, $ 1002.0 (group Special)
```

Figure 13. Console Users Window for a Selected User

Seeing a Chronological User Log

To see a chronological log of console use by each user, Dan clicks the **Console Log** button. A Console Log window, similar to Figure 14, opens.



```
root      Tue Jun  2 1998  07:59 - 08:33  33
writer1   Tue Jun  2 1998  08:33 - 08:34  0
dans      Tue Jun  2 1998  08:34 - 08:34  0
writer1   Tue Jun  2 1998  08:35 - 09:03  28
writer1   Tue Jun  2 1998  09:03 - 09:20  16
spinguy   Tue Jun  2 1998  09:20 - 09:21  0
root      Tue Jun  2 1998  09:22 - 11:16  114
spinguy   Tue Jun  2 1998  11:17 - 11:31  14
root      Tue Jun  2 1998  11:32 - 11:34  2
writer1   Tue Jun  2 1998  11:45 - 11:59  13
dans      Tue Jun  2 1998  11:59 - 12:00  0
writer1   Tue Jun  2 1998  12:00 - 14:56  175
spinguy   Tue Jun  2 1998  15:01 - 15:02  1
writer1   Tue Jun  2 1998  15:02 - 08:28  1045
root      Wed Jun  3 1998  08:28 - 09:03  35
root      Wed Jun  3 1998  09:05 - 09:09  4
vnmr1     Wed Jun  3 1998  09:10 - 09:10  0
vnmr1     Wed Jun  3 1998  09:10 - 09:51  40
nmrguy    Wed Jun  3 1998  09:52 - 09:53  0
```

Figure 14. Console Log Window

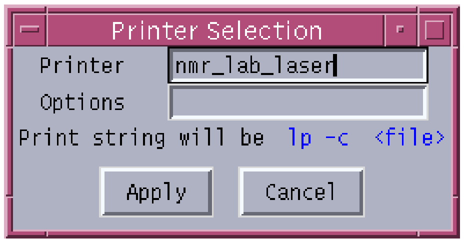
Seeing and Printing Reports

To see and print individual summaries for each group, Dan clicks the **Print all groups** button.

To print a summary of individual charges, Dan clicks the **Print main window** button.

Selecting A Printer

To select a printer for reports, Dan does the following steps:



Printer nmr_lab_laser
Options
Print string will be lp -c <file>

Figure 15. Printer Selection Window

1. From the **File** menu, he selects **Printer** to open a Printer Selection window similar to Figure 15.
2. In the **Printer** field, he enters the name of a printer known to his system. The default is `lp`.
3. In the **Options** field, he enters print options (if any).
4. He clicks on **Apply**. Dan's printer selection remains current until he quits the program.

Dan can also do the following procedures to further administrate users and groups.

- "Deleting a Group," next
- "Deleting a User from a Group," next
- "Updating and Saving Records," page 83

Deleting a Group

To delete an empty accounting group, Dan does the following procedure.

1. From the **Edit** menu, select **Delete_Group**.
2. In the **Group name** field, select the group to be deleted.

   If the group has members, you must delete them from the group first.
3. Click the **Apply** button.

Deleting a User from a Group

To delete a user from an accounting group, Dan does the following procedure:

1. From the **Edit** menu, select **Delete_User**.
2. In the **Group name** field, enter the name of the group from which the user is to be deleted.
3. In the **User name** field, enter the name of the user to be deleted from the group.
4. Click the **Apply** button.

Updating and Saving Records

Updating and saving records is an administrative option available *only* to `root` users. This feature prevents users other than `root` from editing accounting records. To update and save records, Dan does the following procedure:

1. He logs in as **`root`**.
2. He clicks the **Update & Save** button.

   The information is saved in the `/var/adm` directory as the file `wtmp.mm.dd.yy` and the accounting logs are reset.

Exiting the Accounting Program

To exit the program, Dan can either click the **Exit** button OR select **File** from the main menu, then **Quit**.

Navigating the VNMR Accounting Tool

The VNMR Accounting tool, shown in Figure 6, consists of a menu bar at the top, two panes that list groups and unaccounted users, and two rows of buttons at the bottom.

Menu Bar

The top of the window has a menu bar with the following options, each of which can be clicked to show a submenu:

File	For selecting a printer and printing options or for exiting from the accounting program.
Edit	For adding groups and users or for deleting groups and users.
Holidays	For assigning holidays for billing purposes.

Lists of Groups and Unaccounted Users

The middle of the window has two panes listing accounting groups and unaccounted users:

Defined Groups	The upper pane shows a list of accounting groups defined by the program, with the amount of time and the charges for each group. If all users have been added to a group, only this list appears.
Unaccounted Users	The lower pane lists users who have not been added to a group.The first time you run the accounting program, the accounting window lists only unaccounted users , similar to Figure 6, because no groups have yet been added. The first column identifies the VNMR user. The second column lists the total amount of time, in minutes, that user was logged on to the system. Use of VNMR by unaccounted users is reported as time-only because these users have no cost structure associated with them.

Buttons

The bottom of the window has two rows of buttons:

Recalc	Recalculates all accounts.
Update & Save	Saves all accounting information in the directory `/var/adm` and resets accounting logs.
Print All Groups	Prints a summary of charges for each group.
Print Main Window	Prints a summary of all charges.
Exit	Exits the accounting program.
Console Users	Displays a list of console users. Double-clicking on a line in the display opens a detailed window showing the user’s activity.
Console Log	Displays a window of system use, sorted chronologically.

Chapter 4. Output Devices, Terminals, Serial Ports

Sections in this chapter:

- 4.1 "/vnmr/devicenames File," this page
- 4.2 "/vnmr/devicetable File," page 86
- 4.3 "Printing and Plotting under VNMR," page 87
- 4.4 "UNIX Aspects of Printing and Plotting," page 88
- 4.5 "Null Modems for Printers and Plotters," page 89
- 4.6 "Fixing Problems with Printing and Plotting," page 90
- 4.7 "Connecting Serial Terminals," page 90
- 4.8 "Using Extra Ports," page 91
- 4.9 "Using an I/O Port for Acquisition Diagnostics," page 91

This chapter describes the functionality of the VNMR and UNIX printing facilities. In most cases, printer installation is achieved using easy-to-use graphical utilities, and *the detailed information provided in this chapter is usually not required for installing and using printers and plotters under UNIX and VNMR*. However, for debugging in the case of printing problems or for the installation in non-standard cases (printers on extra ports, non-standard printers, certain remote printers, even printing on non-standard paper formats), it is advantageous to know how to read and modify the administration files directly.

Most commands mentioned in this chapter must be performed by the UNIX system administrator `root`. Should you not have `root` permission on your system, you must involve the system administrator in these operations.

4.1 /vnmr/devicenames File

The text file `/vnmr/devicenames` defines printers and plotters for use with VNMR. In all cases, this file must be edited after loading it from the original software. This is usually done using the `adddevices` utility, see below. However, if you are just loading a new version of VNMR, and the Solaris installation remains untouched, you can reuse the `devicenames` file from the previous version of VNMR.

Even though you may not edit `devicenames` directly, it is sometimes useful to be able to be able to interpret the contents of this file in case of problems with printing or plotting. Each printer or plotter must be defined in this file with the following entries:

- `Name` – Can be anything. Value can be different from `Type`.
- `Usage` – Can be `Printer` or `Plotter`. Under Solaris 2.x `Both` should not be used.
- `Type` – A selection from a list given in the comment.
- `Host` – The name of the host to which the device is hooked up.

- `Port` – `/dev/ttya` or `/dev/ttyb` for devices connected to a serial port, `/dev/bpp0` for the parallel port. Some network printers (printers connected to Ethernet) also use a pseudo port entry.
- `Baud` – `9600` or `19200` for most devices, unused for devices on the parallel port.
- `Shared` – Usually set to `No`, see comment section.

Any device may be defined several times under different names (e.g., `HP7550` and `hp`), although usually it is preferable to use a single entry with a short name indicating what the device is (in networks, the name may also indicate where the device is located). Some devices might be defined with different characteristics (different resolutions for matrix printers, devices in “reversed” mode). Changing the printer or plotter by menu later is easier if a minimum number of options are activated.

For example, for a LaserJet printer (which can also be used as a plotter), define one entry (typically `Laserjet_300R`) for standard quality plotting (Usage: `Plotter`) and perhaps one other entry (typically `LaserJet_150R`) for rapid output (Usage: `Plotter`). Leave out the portrait format options for plotting because they are rarely used.

For printing, it is recommended to have the standard printer defined with the name `lp`, because that will make life easier later on (with a LaserJet, take any unused entry, name it `lp`, and set the usage to `Printer`). `lp` is the default device with the `vnmrprint` shell script. If `lp` is a defined printer, the printer name does not have to be specified when printing out files in UNIX using `vnmrprint`.

The end of the comment can be found easily in `vi` by searching for the expression `COMMENTEND`.

Even if you don’t have a PostScript or HP/GL plotter, you may still want to have such devices defined in `/vnmr/devicenames`. This allows you to select such a device in VNMR for plotting into a file. Use `page(filename)` in VNMR, instead of just `page`, and make sure you don’t incidentally send data to a non-existent printer!

4.2 /vnmr/devicetable File

The file `/vnmr/devicetable` defines the characteristics of all possible printers and plotters. It defines attributes such as steps per mm, character size, offsets, and drawing limits. If “artificial shoulders” are produced (perhaps by using extra-thin pens) with plotters, the number of steps per mm might be increased (e.g., change the parameter `ppmm` from 5 to 10) for smoother plotting output. In such a case, note that the character size (`xcharp1` and `ycharp1`, in steps) must be increased accordingly; otherwise, text output shrinks in size.

The default character size can also be changed easily in the `devicetable` file. Make a backup copy of the original file before you start changing it. After changes are made to the `devicetable` file, the plotter must be selected again to enable the changes. This can be achieved by setting `plotter=plotter` in VNMR, which reenters the current plotter and causes VNMR to reread the entry in `/vnmr/devicetable`.

Most entries in this file are quite obvious. The `Printcap` entry is not used with SVR4 (`lp`) printing. For VNMR, the entry `raster` defines the type and orientation of the plot output: `0` is HP-GL output, `1` is PCL (raster, used on LaserJet and other dot matrix printers) in “portrait” orientation, `2` is PCL in “landscape” orientation, `3` is for PostScript in “portrait orientation, and 4 is PostScript in “landscape” orientation.

4.3 Printing and Plotting under VNMR

VNMR stores information for printing and plotting in files in `/vnmr/tmp`.

Printing and Plotting Files

For plotting, the file name for such files is composed of four parts without a separator:

- Local host name
- Letters `PLT`.
- Process-ID of the Vnmr foreground process.
- Number that is incremented (starting at 0).

For printing, `PLT` is replaced by `PRINT`.

Some examples of file names:

```
uster1PLT88940
uster1PLT88941
...
uster1PLT889413
uster1PRINT88940
uster1PRINT88941
```

In all these examples, the host name is `uster1` and the VNMR process-ID was `8894`.

vnmrprint Shell Script

For printing, VNMR does not send the files straight to the printer (by calling `lp <pr_file>`). Rather, it calls a shell script `/vnmr/bin/vnmrprint` as follows:

```
vnmrprint /vnmr/tmp/pr_file <printer_name> <file>
```

The `printer_name` argument could serve to do something specific to a certain output device type (as defined in `/vnmr/devicetable`) in the shell script. At the same time, this argument causes `vnmrprint` to delete the file after printing it. The `file` argument is used to indicate a disk file into which the output should be captured (rather than sending it to the printer). If the last argument is `clear`, this indicates that the accumulated printing information is to be discarded. VNMR uses this with the command `page('clear')`.

Let's have a look at the shell script `/vnmr/bin/vnmrprint` (actually a Bourne shell script, see "Bourne Shell Scripts," page 212). The central line in that file could be written as follows:

```
expand $1 | sed -e 's/^/    /' | fold | lp -s -o nobanner-d $2
```

In this part of the script, `$1` is the name of the file to be printed and `$2` is the name of the printer. The print data are sent through a number of separate programs before they are sent to the printer: `expand` replaces `tab` characters by spaces (8 spaces each—the printer does not "know" the standard UNIX tab settings), `sed` inserts 6 spaces at the beginning of each line (the "hat" sign "^" between the slashes indicates the beginning of each line).

The `fold` command in this script folds over lines longer than 80 characters (otherwise, characters behind column 80 would be lost with certain printers). The line lengths can be specified (e.g., using `fold -88`). With this setup, normal lines start at column 7 and the remainders of folded lines start at column 1. If all lines should start at column 7, the commands in the script would have to be rearranged as follows:

```
... | fold -74 | sed -e 's/^/    /' | ...
```

Note that `fold` now folds after column 74, because `sed` is going to insert blanks after folding.

Finally, `lp` sends the data to the output device (or, more correctly, to the spooling software). The `-s` option causes `lp` to suppress printing messages; the `-o nobanner` option suppresses the header (“burst”) page.

The lines shown above from `/vnmr/bin/vnmrprint` appear four times:

- For the case where only the file name is given in the argument, a standard printer named `lp` is assumed, and the file is not deleted after printing.
- If two arguments are given, the file is sent to the specified printer and is not deleted after printing.
- If three arguments are given for other printers, the file is sent to the specified printer, and is deleted after printing.
- If four arguments are given, the print data are sent to a file (indicated in the fourth argument) instead of being plotted. If the fourth argument is `clear`, the data are discarded (no plot, no file).

The modes with two, three, or four arguments are the only ones used by VNMR. The first mode, only a file name, can be used for calls from within a shell (e.g., a UNIX window). Make sure that from within VNMR, `printon dg printoff` prints correctly. If you want to change `vnmrprint` for interactive use, you only need to change the first (if `lp` is a defined printer) or second occurrence of the line shown above.

Why is printing under VNMR so complicated? If you send a print file directly to `lp`, you have no formatting control whatsoever (unless you explicitly pipe the data through a filter program). The idea behind `vnmrprint` is to have some minimal treatment that would ensure proper printing of unformatted documents but not severely affect the formatting of preformatted documents.

vnmrplot Shell Script

For plotting, a shell script `vnmrplot` is used instead of `vnmrprint`. `vnmrplot` contains a simple call to `lp` of the type

```
lp -s -c -o nobanner -d $2 $1
```

where `$2` is the name of the plotter, and `$1` is the file to plot. The `-o nobanner` option omits the header (banner) page, the `-s` option causes lp to suppress printing messages, and the `-c` option makes `lp` place a copy of the data in the spool directory, rather than just creating a symbolic link and plotting from the original file itself (in which case, it would be quite likely that `vnmrplot` would delete the file before plotting is finished).

Plot files can be quite large. A 5-MB file is not unusual for HP-GL or Postscript 2D plots with many signals and contour levels. Therefore, be sure there is plenty of free space in `/var` and `/tmp`. Similar to `vnmrprint`, an additional argument to `vnmrplot` permits “plotting into a disk file.”

4.4 UNIX Aspects of Printing and Plotting

There are two aspects to using printers under UNIX: the user aspect (the actual print command, and commands to cancel print requests and check scheduled requests) and the aspect of printer setup and administration. the UNIX aspects of setting up new printers is covered in the *VNMR and Solaris Software Installation* manual. In the following sections, we want to give you some more insight in the Solaris printing software, to enable you to find and fix eventual printing problems.

How to Use Printers Under Solaris

The command to print under Solaris is `lp`. The data to be printed can either be specified as a file name:

```
lp -d printer_name filename
```

Alternatively you can use a UNIX pipe to print the output of a UNIX command:

```
ls -l | lp -d printer_name
```

This can also be used for formatting a text file prior to sending it to the printer:

```
cat filename | fmt -c -80 | lp -d printer_name
```

The `fmt` command is a simple text formatter.

If you have defined your printer as `lp`, then you don't need to specify the printer:

```
lp filename
cat filename | lp
```

Even if your printer is not named `lp`, you can still define it as default device to the `lp` command by defining an environment variable LPDEST:

```
setenv LPDEST printer_name
```

It is best to define this in your local `~/.login` file. Solaris print requests normally generate a leading header page that indicates who generated the following output, what the file name was (if the file name was specified as argument), date and time when the print request was submitted, and the print job number (see below). If you want to suppress this header page, you can use the `-o nobanner` option:

```
lp -d printer_name -o nobanner filename
```

or, when using a UNIX pipe:

```
cat filename | lp -d printer_name -o nobanner
```

The `lp` command reports the ID for your print request. For example,

```
> lp -d laser filename
request id is laser-23583 (1 file(s))
```

where `laser` is the name of the printer and `23583` is the ID of the print request.

The ID number can be used to cancel specific requests using the `cancel` command, if required (see `man cancel`). The same can be done more easily using the printer tool from the CDE toolbar.

How Does Solaris Printing Work?

In this section, which will be added in the next version of this manual, we plan to give you information on the setup files and the directories involved in the definition of the printers and in the spooling process. This might enable you to locate errors in the case of printing problems.

4.5 Null Modems for Printers and Plotters

Printers and plotters connected to port a or port b on the CPU board require a null modem; however, most RS-232 expansion boards can be configured so that no null modem is required. Often, a null modem cable is used instead of null modem. We use “null modem” because what is commonly used for a null modem is a little box inserted between the cable and one of the connectors the cable connects to.

Although many output devices only use software handshaking (which only connects lines 2, 3, and 7 of the RS-232 cable), it is recommended that you use a proper, full RS-232 null modem (pins 1–>1, 2–>3, 3–>2, 4–>5, 5–>4, 6+8–>20, 7–>7, and 20–>6+8). A null modem can also be devised be rewiring one connector of a cable (clearly mark the cable in this case), but this is not a commonly recommended practice.

Note that the null modem described here is a standard version recommended by Sun for asynchronous serial communication with software and hardware handshaking. There are other standards on the market, such as for PC computers, that do not ensure proper hardware handshaking. It is essential to use a properly configured null modem.

4.6 Fixing Problems with Printing and Plotting

Hints for fixing printing problems will be added to this section in future version of this manual as we receive the information. Most operations must be performed as the UNIX system administrator `root`.

The first and most important hints are the following:

- If printing or plotting on a certain device does not work, it is usually best to delete the entire printer definition in both VNMR and Solaris, and to restart defining that printer from scratch.
- Follow exactly the instructions given in the manual *VNMR and Solaris Software Installation*.
- If you are setting things up “by hand,” with PostScript printers it is absolutely essential to use *separate* definitions for printing and plotting.

4.7 Connecting Serial Terminals

With the advent of X terminals that communicate with the Sun workstation through Ethernet, simple serial terminals (such as GraphOn and Tektronics) has become far less popular. X terminals are not much more expensive than the serial models, and they offer far more comfort to the user as well as full-featured X Window system operation. X terminal emulators are also available for PCs and Macintoshes. More information will be added in a future version of this manual.

Currently, the main use of serial terminals is in the area of remote access through modems. See Chapter 19, “Other Networking Options,” for more information about this topic.

As with printers and plotters, terminals must be connected through a null modem (with some RS-232 expansion boards no null modem is required). The same type of cable or null modem can be used as for printers or plotters, see “Null Modems for Printers and Plotters,” page 89. Again (as for printers and plotters) you should preferably use a full null modem with the following connections: pin 1–>1, 2–>3, 3–>2, 4–>5, 5–>4,
6+8–>20, 7–>7, and 20–>6+8.

4.8 Using Extra Ports

Sun systems can be equipped with additional serial or parallel printer ports, typically by adding an SBus expansion board. For their installation and use, see the associated documentation. To use such a port under VNMR, you may want to set up a printer using a standard port and then modify the port name in the newly generated Solaris printer control files.

4.9 Using an I/O Port for Acquisition Diagnostics

Sun and UNIX provide basic facilities to use a window as acquisition diagnostics terminal using any one of the RS-232 ports, involving the `tip` command and the `/etc/remote` file. At present, however, the result is relatively unsatisfactory because it causes CPU slow-down and other problems. This procedure is not recommended, certainly not as a long-term solution.

Chapter 5. Software Operating Levels

Sections in this chapter:

- 5.1 “Solaris Run Levels,” this page
- 5.2 “Login Shell,” page 99
- 5.3 “Windowless Multiuser Mode,” page 100
- 5.4 “Graphical User Interface,” page 100
- 5.5 “Shutting Down the System,” page 103

UNIX is a complex and powerful operating system. The complexity not only is in the large number of files and directories (Chapter 6, “Disk Organization,” covers this topic), but also in the large number of processes involved (many running simultaneously, in a time-shared mode), in the way these processes are started up, and in their interaction and interdependency. In this chapter, we discuss how to control (i.e., switch between) the various operating levels, trying to give you some insight on what happens when Solaris 2.x starts up and shuts down:

5.1 Solaris Run Levels

There are five different run levels in the Solaris environment:

level 0	Runs at the PROM-based software level on Sun workstations.
level 1	Single-user mode, for diagnostics and system maintenance.
level 2	Multiuser mode, but without networking facilities (processes and daemons relating to networking are not started).
level 3	Normal Solaris multiuser operation, including all networking activities as defined in the administration files.
level 4	This is available as an alternate multiuser operating mode; currently unused.

Level 2 did not exist in SunOS 4.x. It was introduced with the transition to Solaris 2.x (i.e., System V UNIX).

Run Level 0—Monitor Mode

The monitor mode (run level 0) is PROM-based software that serves three main purposes:

- Booting UNIX.
- Performing basic hardware diagnostics.
- Setting the boot parameters (default boot device and options), hardware diagnostics parameters (e.g., defining what is tested automatically upon bootup and how), and basic system setup parameters (character screen dimensions, system startup logo, etc.).

The monitor mode is passed quickly when the system automatically boots up after switching on. This mode is reached either by shutting down UNIX (for rebooting or switching off the workstation) or by interrupting UNIX by pressing Stop-a (pressing the Stop key and A key simultaneously). The monitor mode runs in a white ASCII screen, with no windows, and is characterized with the `ok` prompt. The most relevant commands of this mode are shown below (commands considered more important are shown in bold):

`boot`	Boot up to default run level, from the default boot device
`boot -r`	Boot up from default boot device, check what devices and peripherals are present, add device entries to `/dev` where necessary (used after adding peripherals)
`boot -s`	Boot up from default boot device, into single-user mode only
`boot disk`	Boot up from disk (default)
`boot cdrom`	Boot up from CD-ROM
`boot cdrom - browser`	Boot up from CD-ROM and install Solaris using a Web browser (*only for a system with Solaris 2.6 and higher, at least 48 MB of RAM, and a boot disk with at least 1 GB*)
`boot net`	Boot up via Ethernet (diskless systems only)
`go`	Continue after pressing Stop-a
`help`	Show help on the monitor mode
`help diag`	Show help on PROM-based diagnostics
`help setenv`	Show help on the `setenv` monitor command
`printenv`	Display EEPROM settings (you can also get this information in a shell, when running Solaris, using the `eeprom` command).
`reset`	Reset all hardware, self-test, reboot
`setenv <args>`	Change EEPROM settings
`sync`	Sync disk and reboot after Stop-a

Pressing Stop-a should only be used in emergency cases (e.g., when the user interface is hung and cannot be activated by Ctrl-c, Ctrl-d, pressing the Stop key alone, killing windows, or killing processes from an other system using a remote login). If Stop-a was pressed by mistake, you can type `go` in monitor mode to resume the previous activities (you may need to repair the screen using the Refresh option from the background menu).

CAUTION: **If it was really necessary to use Stop-a, it is strongly recommended that you enter the sync monitor command to continue. sync attempts to synchronize the disk prior to rebooting and may help avoid unnecessary file system inconsistencies and lost or corrupted files.**

Earlier Sun workstations had a different monitor mode that used the “>” prompt and had a very terse syntax (most commands were a single character), but essentially offered the same capabilities as the new mode. The main reason Sun created the new monitor mode was to increase safety. In the old mode it was relatively easy to incidentally change EEPROM locations with “arbitrary” keyboard sequences. The new mode uses much longer commands that make it unlikely that commands are typed by mistake.

Current workstations still have a rudimentary subset of the old monitor mode implemented, but we recommend you use the new mode exclusively if possible. If you happen to get the “>” prompt, you can type `n` to switch to the new monitor mode. The command `old-mode` is used to change from the new mode to the old mode.

CAUTION: **Only if you have reached the monitor mode using the proper procedures (`init 0` or `shutdown`, see below) is it safe to switch off the system using the on/off switch in the back of the CPU. Switching off the system otherwise can corrupt data.**

Run Level 1—Single-User Mode

The single-user mode is the proper operating level for checking disk partitions with `fsck`, for dumping or restoring disk partitions without user interaction, etc. In single-user mode, all daemons that could alter disk files are switched off—there is no real multitasking, no swapping, no network activities, no printing, no logins (other than for `root`), and, of course, no acquisition on spectrometer hosts.

As of Solaris 2.x, the `root` password must be entered in order to gain access to the single-user mode[1]. A C shell is opened after logging in, with a hash sign "#" as default prompt. At this point, you can perform system maintenance tasks such as file system checks, backups, and repairs (see Chapter 12, "File Security, Repair, and Archiving," for further information).

Keep in mind that the single-user mode is meant for system administration only. You may experience certain restrictions in this environment (e.g., the command path includes a limited number of directories so that some commands may require an absolute path).

Once you are finished with your system administration task, you should either shut the system down or continue the bootup to multiuser mode:

- Press Ctrl-d to let the system boot up to its default run level (usually 3).
- Enter `init 6` to reboot the system completely (again, up to its default run level)
- Enter `init 0` to shut the system down to monitor mode (e.g., for switching it off)
- When (and only when) instructed by the `fsck` command, you may interrupt the system by pressing Stop-a to bring it down to monitor level. *Use the with caution: pressing Stop-a can corrupt data on a disk.*

Run Level 2—UNIX Without Networking

In this run level, daemons and processes operating networking activities are not started. One can use this run level to operate the system temporarily as a standalone workstation, for system maintenance tasks (such as backing up disk slices on tape) for which remote activities (such as `rsh`, `rcp`, `rlogin`, `ftp`, `telnet`, etc.), and interference by other users should be suppressed, in order to obtain consistent file system dumps.

For spectrometer hosts (at the very least for UNITY*INOVA*, *MERCURY*-Series, and *GEMINI 2000* systems) at this run level, acquisitions are not possible, the same as in single-user mode. Therefore this run level is of little use for most VNMR users and spectrometer system administrators.

[1] This is an essential security feature. Under SunOS, any user could shut the system down by pressing Stop-a, boot into single-user mode, and the break into the system, remove passwords, etc. On the other hand, you must now not forget the `root` password if you don't want to risk losing system administration access.

Run Level 3—Networking Included

This is the standard, default UNIX and Solaris operating mode; logins by other users are possible both locally as well as remotely. When reaching that run level, a login process is started on the main terminal. With CDE installed, this login shell is normally superseded by an `xdm` graphical login session that gives access to either OpenWindows or CDE.

init and shutdown Commands

The `init` command is used to switch between the various operating levels:

`init 0`	Run system down to monitor mode (run level 0).
`init S` **`init s`**	Switch to single-user mode (run level 1), for diagnostics and system maintenance (`S` and `s` are identical).
`init 2`	Switch to multiuser mode without networking facilities (run level 2).
`init 3`	Switch to multiuser mode (run level 3).
`init 4`	Switch to alternate multiuser mode (run level 4, not implemented).
`init 5`	Run system down completely, power off (where possible).
`init 6`	Run system down to monitor mode, and reboot it to its default run level (usually run level 3).

The most frequently used `init` options are `init 0` to run the system down to monitor mode (`halt` in SunOS 4.x) or `init 5` to for switching it off completely and automatically powering it down where possible (most users just use `init 0`), `init 6` for rebooting the system (`reboot` in SunOS 4.x), and occasionally `init s` for entering the single-user mode.

`init` is also used by the `shutdown` command, which can be used to shut down the system more gracefully to the users. `shutdown` first warns all users about the upcoming system shutdown, then, after a “grace period,” the system is shut down using the `init` command. The default grace period is 60 seconds, and by default, `shutdown` runs the system down to single-user mode

`shutdown`	Run system down to single user mode.
`shutdown -g300`	Same, but with a 5-minute grace period.
`shutdown -g300 ″maintenance shutdown″`	Same, with message

With the `-i` option, `shutdown` can be used to reboot the system or to run it down completely:

`shutdown -i6`	Reboot system, with 1-minute grace period.
`shutdown -g300 -i6`	Same, but with a 5-minute grace period.
`shutdown -g300 -i0 ″maintenance shutdown″`	Run system down completely, with warning message for users and 5-minute grace period.

On a multiuser system, the command `shutdown -i0` is preferred for shutting down a system. On a system with only a single user, you can also safely use the `init 0` command.

The SunOS 4.x `fasthalt` and `fastboot` commands have been eliminated. They are no longer required, because Solaris 2.x only checks file systems at bootup, when it is really necessary.

How Does the UNIX Bootup and Shutdown Work?

Under SunOS, the system startup was done by a small number of shell scripts: `/etc/rc`, `/etc/rc.boot`, `/etc/rc.local`, `rc.ip`, and `rc.single`. In Solaris 2.x, the task of booting up the system and shutting it down is done with several shell scripts, depending on the target run level. The principal scripts (`rc0`, `rc1`, `rc2`, `rc3`, `rc5`, `rc6`, and `rcS`) are stored in `/sbin`. In order to understand what these shell scripts do, let us look at the central part of `/sbin/rc3`:

```
if [ -d /etc/rc3.d ]; then
    for f in /etc/rc3.d/K*
    {
        if [ -s ${f} ]; then
            case ${f} in
                *.sh)   .        ${f} ;;        # source it
                *)      /sbin/sh ${f} stop ;;   # sub shell
            esac
        fi
    }
    for f in /etc/rc3.d/S*
    {
        if [ -s ${f} ]; then
            case ${f} in
                *.sh)   .        ${f} ;;        # source it
                *)      /sbin/sh ${f} start ;;  # sub shell
            esac
        fi
    }
fi
```

The `rc3` script looks at the files in `/etc/rc3.d` (each of the run levels has its associated directory in `/etc`: `rc0.d`, `rc1.d`, `rc2.d`, `rc3.d`, `rcS.d`). In a first `for` loop, `rc3` looks at files starting with *K*; each of these files is executed in turn, *in alphabetical order*. The way the scripts are called depends on the name of the script. If it has a `.sh` extension, it is simply executed; otherwise, it is called with an argument `stop`.

In a second `for` loop, `rc3` does the same thing with all files starting with an *S* in the same directory, except that entries without `.sh` extension are now called with an argument `start`. `rc3.d` contains just five entries:

```
S15nfs.server
S31smc.init
S31smc.init
S77dmi
S99TAS
```

There are no *K* files, just five *S* (startup) files.

After the first character, which determines whether a script is a `start` or a `kill` script, the name contains a two-digit number, followed by some descriptive part that indicates what that script deals with. The numeric part determines the sequence in which the scripts are called.

Many things in a bootup sequence must happen in a certain order (e.g., certain programs can only be started once some specific other programs have been launched), and that order is controlled simply by the numeric part of the script name. To change the order of execution, the script must be renamed appropriately.

Obviously, these five scripts are not the entire story for starting up multiuser mode. Prior to executing `rc3`, the script `rc2` is called, which operates on the files in `rc2.d`. It turns out that `rc3.d` is the smallest of the `rc*.d` directories. All the others contain many more files: 19 scripts in `rc0.d` and `rc1.d`, 40 in `rc2.d`, 11 in `rcS.d`. The scripts `rc0`, `rc5`, and `rc6` are all identical and use the files in `rc0.`

Which one of the scripts that is called by which `init` option is defined in the file `/etc/inittab`. `init` does more than just to call the `rc*` scripts—see `man inittab` for more information.

Overall, there seems to be a confusing number of shell scripts involved in the system startup and shutdown; however, most of the files in the `rc*.d` directories are just *hard links* to files in a directory `/etc/init.d`. Here is a listing of that directory:

```
ab2mgr                  inetsvc          RMTMPFILES
acct                    init.dmi         rootusr
ANNOUNCE                init.snmpdx      rpc
asppp                   initpcmcia       rtvc-config
audit                   keymap           sendmail
autofs                  leoconfig        skipes
autoinstall             lp               skipkey
buildmnttab             mcd              spc
buttons_n_dials-setup   mkdtab           standardmounts
cachefs.daemon          MOUNTFSYS        sysetup
cachefs.root            networker        sysid.net
cacheos                 nfs.client       sysid.sys
cacheos.finish          nfs.server       syslog
cron                    nscd             TAS
devlinks                pcmcia           ufs_quota
dhcp                    perf             utmpd
drvconfig               power            uucp
dtlogin                 PRESERVE         volmgt
inetinit                README           xntpd
```

Each of these scripts deals with a specific task in the operating system. Let’s just pick one example: `init.d/volmgt` handles the volume management (automatic recognition and mounting of floppy and CD-ROM). There are two extra hard links to this file: `rc0.d/K73volmgt` and `rc2.d/S92volmgt`. Because `K73volmgt` has a name starting with *K*, it is run with the argument `stop`. Whereas the `S92volmgt` script in `rc2.d` has a name starting with *S*, so it is called with the argument `start`. In other words, `init.d/volmgt` is a shell script that starts *and* stops the volume management:

`init.d/volmgt start`	Starts volume management
`init.d/volmgt stop`	Stops volume management

By placing hard links (these could also be symbolic links) with the appropriate names in the specific `rc*.d` directories, we can determine for which run level volume management must be started, where it must be stopped, and where exactly in the bootup sequence.

Similarly (just to give you a second example), the script `init.d/lp` has three extra links: `rc0.d/K20lp`, `rc1.d/K20lp`, `rc2.d/S80lp`. In other words, the printer software is stopped both for run level 0 and for run level 1 (single-user mode), and started in run level 2 (the first step in starting up multiuser mode or run level 3).

Overall, these shell scripts perform much the same as the `rc*` scripts in SunOS 4.x or BSD UNIX for starting up the operating system. The main difference is that in Solaris 2.x not only the system startup is performed in an orderly and controlled manner, but also the system shutdown is performed in a sequence that avoids unwanted side effects (e.g., by stopping the processes in random or uncontrolled sequence and killing resources that might still be used by some other program to terminate properly).

5.2 Login Shell

Once the multiuser mode is reached, there is a process `ttymon` that watches out for active terminals, primarily the main keyboard and user interface. For each active terminal, `ttymon` starts a `login` process (on serial terminals, a return character must be given to make `ttymon` start the `login` process).

The login process asks for the user name. If a password is defined, it asks for the password, and then compares the user entry with the encrypted password in `/etc/shadow`:

- If the password matches, the login process puts the user in his or her home directory, defined in `/etc/passwd`, and starts up the initial shell, also defined in `/etc/passwd`. The shell is normally a C shell, `/etc/csh`. In this case, `~`*`username`*`/.cshrc` is executed, as well as `~`*`username`*`/.login`.
- If the login process does not find the user name in `/etc/passwd`, it asks for a password anyway, to make it more difficult to outsiders to find out defined user names.

In Solaris 2.x, a login process is also started in single-user mode. This blocks a security hole in SunOS 4.x, where a system could simply be aborted and brought up into single-user mode, in which passwords could be removed.

It is strongly recommended to have passwords defined for every user, particularly in networked or open-access situations. A good password should not just contain alphabetic characters:

- It should be mixed (lower and upper) case and include numeric characters.
- The length should be 7 or 8 characters (extra characters beyond that are disregarded).
- It should not be a birthday or a telephone number, and should not be found in any dictionary for any language.
- It should be changed at regular intervals.

Solaris checks the password when it is changed by the user and refuses to accept simple character permutations (reshuffling) of the old password.

Primarily, the login process is started in a white, non-graphic screen. If CDE is loaded, that login screen is superseded by `dtlogin`, a graphical login screen. This not only looks nicer than the traditional text-based login, it also gives the user a choice:

- Log into the CDE environment, see "Common Desktop Environment," page 101.
- Log into the OpenWindows environment, see "OpenWindows," page 102.
- Log into a "fail-safe environment" (a single `xterm` window that cannot be used for running VNMR).
- Select "Command Line Login", the window-less (white, non-graphic) multiuser mode, see "Windowless Multiuser Mode," next.

5.3 Windowless Multiuser Mode

On systems without CDE, the initial environment after the login is the windowless multiuser mode. This offers a single shell interface—typically, a C shell for most users or a Bourne shell for `root`. This mode resembles early, non-graphical UNIX interface. Background processes can be started in order to allow for multiple, simultaneous tasks, but this only makes sense as long as the background processes don't generate substantial amounts of output.

Today, this mode is used for simple system administration tasks or for shutting down the system. Both activities must be done as `root`, and both can also be done in a graphical environment.

For regular users after a login, the shell script `~/.login` is executed, which automatically starts the OpenWindows graphical user interface. The windowless multiuser mode is only reached again after quitting the OpenWindows interface. If the `~/.login` script is modified accordingly, it may even log the user out again automatically, such that the windowless multiuser mode never is active for these users. In order to do this, add the three lines

```
clear         # get rid of cursor rectangle
echo 'Automatically logging out ... '
logout
```

after the line

```
openwin -noauth
```

in the `~/.login` script.

If CDE is loaded, the windowless multiuser login becomes one of the startup options. If selected on a system running VNMR, this option causes the `~/.login` script to be executed, which then automatically starts up OpenWindows.

5.4 Graphical User Interface

All modern desktop computers and graphics workstations use a graphical user interface (GUI). Solaris 2.5 and later offers a choice of two graphical user interfaces: OpenWindows, Sun's original X Window system user interface, and CDE (Common Desktop Environment), a new, emerging standard for graphical user interfaces, also based on X.

The GUI is the main platform for user interaction with UNIX. Because the GUI features multiple windows, it permits running many (dozens, at least) of applications at the same time and on the same user interface. This is made possible both by the UNIX multitasking and time sharing capabilities, as well as by the high-performance CPUs used in modern workstations.

The graphical user interface primarily is a software layer, on top of which many processes can run in parallel (i.e., at the same level). In addition to that, new processes and graphical utilities can also be started *within* any of the utilities (primarily shells) that run on the GUI layer. This also allows building *hierarchical* process families on top of the GUI layer. VNMR is one example of software that forms its own hierarchy of processes, see "VNMR," page 103.

Common Desktop Environment

The Common Desktop Environment (CDE) is a standardized graphical user interface that was defined and adopted by Sun, HP, IBM, and Novell. CDE is simple and intuitive to use. Because it is based on the Motif X Window system standard, it resembles the Windows user interface used on PCs. Although there are fundamental differences between OpenWindows and CDE, for the user the primary difference between the two user interfaces is in the way windows and their utilities (buttons, menus, scroll bars, etc.) look and work.

The primary interaction with CDE is done through the CDE toolbar at the bottom of the screen, as well as through a background menu, similar to OpenWindows. It is simple for the user to configure the CDE environment. Even the CDE toolbar can be reconfigured easily, and just by using the mouse. Similarly, the screen background and the window frame colors can be set by the user, through mouse clicks

There is little if any need to work with text configuration files for CDE (most configuration files reside in `~/.dt`). Therefore, we will not discuss these here.

Unlike OpenWindows, CDE also provides a way to properly terminate running applications when logging out. Whatever utilities are running at logout, they are restarted when entering CDE the next time. As of version 6.1A, VNMR is fully CDE compliant. You can leave VNMR running when exiting CDE, and it will restart, even in the same experiment, when you re-enter CDE.

CDE has a full set of desktop utilities (file manager, calendar manager, clock, performance meter, shell tools, etc.), including a CDE shell tool, called `dtterm`, and a full-screen text editor, `dtpad` (see also "Text Editors dtpad and textedit," page 141). Besides that, all of the OpenWindows deskset utilities are also available from the program manager in the CDE toolbar; they can even be added to the pop-up menus in the CDE toolbar, simply by dropping the corresponding OpenWindows tool icon into the drop box of the appropriate toolbar menu.

A prominent new feature compared to OpenWindows is that CDE allows switching between any number of desktops. The default is four desktops. The advantage here is that several full-screen applications can be running simultaneously, such as VNMR, Pulsetool, the online manuals, a Web browser, and so on, that would otherwise create considerable window overlap or would require the user to temporarily iconify those applications not needed at the moment.

This last CDE feature may also create problems. Most workstations have 8-bit graphics, which allows for up to 256 colors at any given time. Because of the temptation to open several applications simultaneously, color flashing problems may become more frequent when switching to CDE. This is particularly true for VNMR users, because VNMR uses a fairly large color table. The following hints may help in solving this problem:

- Start VNMR first. This way, it has the correct colors.
- Web browsers are color hogs. Don't use Web browsers (particularly with color images) extensively while running VNMR, but rather call them on a case-to-case basis (maybe when not temporarily running VNMR).
- Limit the color cube size with Adobe Acrobat Reader for VNMR Online, see "Color Flashing Problem with Adobe Acrobat Reader," page 173.
- Open the Style Manager from the CDE toolbar, then select "Color", click on "Number of Colors" and select "More Colors for Applications".

Of course, you can still interact with UNIX through a shell and command line interface, see also "Shell Tool, Command Tool, Dtterm, and Xterm," page 102.

OpenWindows

OpenWindows is Sun's X Window system environment. Initially with the Solaris 2.x operating system, OpenWindows was the standard window interface for Sun workstations. In the meantime, it is being replaced by the CDE graphical user interface.

The OpenWindows environment is usually entered from the graphical login or after using the command line login (through the `~/.login` script). For proper operation, VnmrX requires that environment variables such as `OPENWINHOME`, `LD_LIBRARY_PATH`, and `DISPLAY` are defined properly. The script `~/.login` does this with lines such as the following:

```
setenv OPENWINHOME /usr/openwin
setenv LD_LIBRARY_PATH $OPENWINHOME/lib
setenv DISPLAY hostname:0.0
```

The directories `/usr/openwin/bin` must be inserted into the `path` variable for the command path. This is also done in the `~/.login` script, which then calls `openwin` (the program name for OpenWindows).

The `openwin` shell script defines most of the environment for OpenWindows. It starts up the windowing environment by a call to `/usr/openwin/bin/xinit`. This program calls a shell script `~/.xinitrc`, which loads the user-specific X11 defaults from `~/.Xdefaults`, starts the OpenLook window manager (`olwm`), and calls the configuration file `~/.openwin-init`.

The file `.openwin-init` defines what applications are started automatically when entering OpenWindows (such as VnmrX), and how (fonts and colors, except for those defined through `~/.Xdefaults`, window sizes and labels, etc.). The `rootmenu` is defined through a file `~/.openwin-menu` (see Chapter 13, "UNIX System Customization," for more information).

The console window takes all system error messages as well as messages from VNMR acquisition programs such as `Autoproc` and `Acqproc`. The console window should not be deleted; otherwise, these messages are lost or are dumped into the windowless multiuser screen, which messes up the graphical screen. If the console window is deleted, call a new one from the `rootmenu` (if this is defined in `~/.openwin-menu`) or by the command `shelltool -C&` in any window other than VNMR. If the screen needs to be cleaned up again after this, click on Refresh from the Utilities choice in the background menu.

Shell Tool, Command Tool, Dtterm, and Xterm

Even with a modern, graphical user interface like CDE or OpenWindows, many users (particularly system administrators) may still want, at least occasionally, to interact with UNIX through a command line interface, i.e., through a UNIX shell (usually a C shell). From within each of these shell interfaces, new processes can be started, including new shell tools, forming a hierarchical process family. Note that if these processes run in foreground, and you end the parent process (shell) or window (shell tool), you also end the child process (or the entire family of child processes attached to the one you terminate.

Shell windows come in different flavors. Each has slightly different properties. Most are scrolling (dtpad, xterm, and command tool), but shell tool is not. They all behave differently at UNIX and text terminals. Therefore, they all require a different setting of the `term` variable (`TERM` in the Bourne shell), as shown in Table 8.

Under CDE, most users use dtterm, the proper CDE shell window. However, users of the `vi` editor (see "Text Editor vi," page 142) may find that (especially when running `vi` remotely, and in particular when running `vi` under CDE, but off a SunOS 4.x system), that the OpenWindows command tool (`cmdtool`) is more compatible with `vi`, see also "Problems with Running vi," page 145).

The OpenWindows shell tool and command tool are actually one and the same program. You can toggle between the two by enabling and disabling the scrolling. Even with this, you must still make sure that the `term` variable is set correctly after starting up the tool. Note that particularly when doing a remote login, the `term` variable is usually incorrect (`xterm` instead of `dtterm` or `sun-cmd` for standard scrolling windows). If you want to use `vi` remotely, you should first set this environment variable, e.g.:

```
set term=sun-cmd
```

To quit from a shell window, you terminate the shell that runs in that window by using the command `exit` or by typing Ctrl-d.

`xterm` offers somewhat less comfort than the other shell windows. It is primarily used in an X terminal environment, particularly from PC and Macintosh X servers, see also "Using a Personal Computer as an X Server," page 254.

Table 8. Shell Tools and Their Properties

Tool	*Command Path*	*Setting of* `term`
dtterm	`/usr/dt/bin/dtterm`	`dtterm`
command tool	`/usr/openwin/bin/cmdtool`	`sun-cmd`
shell tool	`/usr/openwin/bin/shelltool`	`sun`
xterm	`/usr/openwin/bin/xterm`	`xterm`

VNMR

Within the graphical user interface, VNMR is just one of several active processes, with its own set of windows and graphical utilities. VNMR actually is an entire family of programs and includes several user interfaces (VNMR, *GLIDE*, Pulsetool, Imagebrowser, Acqstat, Acqi, Enter, etc.). Most of the user interaction with this family of programs is through menu buttons and graphical input, but VNMR also has its own command line interface, and its own command ("VNMR shell") language, MAGICAL.

In OpenWindows, VNMR is started automatically when entering the graphical user interface. You can exit from VNMR through the main VNMR menu or using the command `exit`/ VNMR (VnmrX) can be restarted from the OpenLook background menu.

Before exiting the OpenWindows environment, it is important to exit from VNMR first. Otherwise, VNMR is not terminated properly, possibly resulting in corrupted or lost data, and locked experiments.

In CDE, VNMR can be started from the CDE toolbar at the bottom of the screen. Unlike OpenWindows, it is possible to exit CDE while VNMR is still running (starting with VNMR 6.1 only). When CDE is started again next time, VNMR will then automatically start up again (it will even rejoin the experiment that was used before logging out).

5.5 Shutting Down the System

At times, it may be necessary to shut down the entire system or the acquisition cabinet.

To Shut Down the Entire System

Shutting down the entire system should not happen frequently, except when the main power has to be switched off for service operations or other reasons. Use the following procedure:

1. Stop acquisition and then exit VNMR.
2. Exit the windowing environment and log out.
3. Log in as `root`.
4. Shut down the host with the command **`shutdown -i0`**.
5. When the monitor level is reached, shut down external peripherals (disks, tapes, and CD-ROM).
6. Shut down the Sun computer and screen.
7. Shut down the spectrometer console.

To Shut Down Acquisition Computer or Acquisition Cabinet

Shutting down the acquisition computer or the acquisition cabinet may be necessary when boards have to be removed or exchanged from the acquisition computer or when service operations require removing rf boards.

1. Use one of the following procedures to stop the acquisition daemon:
   - If `makesuacqproc` was run during installation, in a UNIX window, enter:
     ```
     su acqproc
     ```
   - If this does not work, open up a UNIX shell window and enter:
     ```
     su -c /vnmr/acqbin/killacqproc -f
     ```
2. If the host computer is to be used for awhile without acquisition, it is best to remove the acquisition flag by entering:
   ```
   rm /etc/acqpresent
   ```

To restart the acquisition computer, you need to get the acquisition daemon running again.

1. Use one of the following procedures:
   - If `makesuacqproc` was run during installation, in a UNIX window, enter:
     ```
     su acqproc
     ```
   - Otherwise, enter:
     ```
     su -c /vnmr/acqbin/startacqproc
     ```
2. If the file `/etc/acqpresent` has been removed, create it again by entering:
   ```
   touch /etc/acqpresent
   ```

Pressing Stop-a (or L1-a on older keyboards) is a hard interrupt or crash. These key combinations should be used only in an emergency and with caution. If you have to use such a hard interrupt, you should afterwards enter `sync` if in new monitor mode or enter `g0` (g-zero) if in the old monitor mode. For details, see "Run Level 0—Monitor Mode," page 93.

CAUTION: **Never have the acquisition daemon running when the acquisition computer is not properly operating. Never switch off or reset the acquisition computer without killing the acquisition daemon first. If such a situation occurs, the acquisition process can severely block or slow down all system activities.**

Chapter 6. Disk Organization

Sections in this chapter:

- 6.1 "Disk Partitioning," this page
- 6.2 "Solaris Directory Structure," page 109
- 6.3 "Multiuser Setup for VNMR," page 117

This chapter describes partitioning and the file system organization for system hard disks

6.1 Disk Partitioning

UNIX allows hard disks to be divided into slices (partitions) that have different functions. To a large degree, these disk slices are treated like different hard disk drives—they are a kind of "logical disk drive." Disk partitioning allows adjusting the size of the various slices to fit user's needs. This partitioning is to a large extent defined when installing Solaris at the initial system setup, or after adding or reformatting a disk (for more information, refer to the manual *VNMR and Solaris Software Installation*).

Naming Disk Slices

Every disk slice is treated as a separate device. Thus, each of the slices in `/dev` has a device entry. Typical slice names are `/dev/rdsk/c0t3d0s0`, `/dev/rdsk/c0t3d0s6`, and `/dev/dsk/c0t0d0s2`. The subdirectory name indicates whether the disk is to be addressed as character device or raw device (`/dev/rdsk`) or rather as block device (`/dev/dsk`)[1]

The naming of these device entries reflects the hierarchical organization of the SCSI disks, and includes the following numbering scheme:

- *controller (*`c`*) ID* – 0 is the built-in (SCSI or EIDE) controller, additional (SCSI) controllers (SBus or PCI cards) receive higher IDs (1, 2, 3, etc.).
- *target (*`t`*) address or ID* – Typical target IDs are 0, 1, 2, and 3 for standard SCSI or SCSI-2 disks. Higher ID numbers can be used if wide SCSI-2 (SCSI-3) or UltraSCSI controllers are used.

[1] The difference between raw and block devices is in the way a program reads or writes from (or to) these devices. With some devices this is dictated by the properties of the device, but mostly it is just a question of what is best for a program. When addressing such devices, a user must know which option to take (both options are defined for hard disk drives) or can find it out by trial-and-error (fortunately, disk device names rarely need to be specified by the user). Some examples: `fsck` requires a raw device to be specified, but with `mount` a block device must be specified.

- *disk* (`d`) *ID* – In theory, a disk controller can control multiple disks; however, all modern SCSI disks come with a built-in (embedded) disk controller. Therefore, in virtually all cases, the disk-ID is 0, because there is only one disk per controller.
- *Slice* (`s`) *number* – Disks can be split into up to 8 slices, numbered 0 to 7.

The disk or slice addresses seem complicated at first—but in reality it is not difficult to correlate devices and their entries in `/dev`. The main complication is with the internal hard disks, for which the SCSI-ID cannot be seen.

- On older Sun systems, the SCSI-ID for these drives was set using a jumper on the disk drive. On newer systems (SPARCstation 4, SPARCstation 5, SPARCstation 20, and Ultra workstations), the SCSI-ID for internal drives is encoded in the internal SCSI connector.
- On the SPARCstation 4, the internal drive is automatically set to SCSI target 3 (`c0t3d0`), which makes the built-in, standard SCSI bus unusable for VXR-S and UNITY spectrometers, where HAL address resides at SCSI target address 3[2].
- On SPARCstation 5, SPARCstation 20, and Ultra workstations, the lower of the internal drives is at SCSI-ID 3 (`c0t3d0`), the upper one at SCSI-ID 2 (`c0t2d0`). On VXR-S and UNITY spectrometer hosts, the internal disk must be switched to the upper slot (SCSI-ID 2), because SCSI-ID 3 is reserved for the HAL. On UNITYplus spectrometer hosts, the upper disk slot (SCSI-ID 2) cannot be used for the same reason.

The software driver for the standard SCSI bus is usually configured for four disks at SCSI targets 0, 1, 2, and 3. External disk drives on the built-in, standard SCSI bus on SPARCstation 4, SPARCstation 5, SPARCstation 20, and Ultra 1/140 systems can use SCSI-ID 0 (`c0t0d0`) or SCSI-ID 1 (`c0t1d0`). On a second SCSI bus, you should use SCSI-IDs 0, 1, 2, or 3 for external disk drives (`c1t0d0`, `c1t1d0`, `c1t2d0`, `c1t3d0`). The fast, wide SCSI-2 (SCSI-3) or the UltraSCSI, such as provided with Ultra 1/170E systems with Creator graphics or newer Ultra (Ultra 30, Ultra 60) systems, allows for a total of 16 devices including, of course, more than four disks[3].

Disk slices are composed of a group of successive cylinder groups. A slice can also cover an entire disk. All partitions on Sun disks feature the fast Berkeley file system called ufs (UNIX file system) under Solaris 2.x. The ufs system is a form of disk organization that not only is reportedly faster than the (more fragmented) System V UNIX file system, but it does not lose as much performance when the disk gets nearly full. The Berkeley file system also is much less subject to aging. Typically, a System V file system gets slower over time, because the files tend to be more and more fragmented (many System V UNIX systems have file system maintenance utilities to clean up and defragment disk partitions).

The fast Berkeley file system performs better mainly because:

- The ufs (BSD) file system uses a bigger block size (8 KB as opposed to 0.5 KB with the System V UNIX file system) combined with variable size fragments (4, 2, 1, and 0.5 KB). A typical file structure consists of fewer elements (but still, small files can be stored efficiently using fragments).
- BSD file systems are organized in cylinder groups (see below), and whenever possible, a file consists of data blocks from the same cylinder group. This not only minimizes the movements of the disk heads (minimizing access times), but it prevents (or at least reduces) scattering of the data blocks of a given file over an entire partition (fragmentation, typical for System V file systems when the disk is nearly full).

[2]. A second SCSI bus resolves this problem.
[3]. There are also SCSI-3 (fast, wide SCSI-2) SBus expansion cards for most Sun systems, and PCI bus UltraSCSI expansion cards for newer, PCI-based Ultra workstations.

For obvious reasons, disk slices cannot be overlapping, but there is an exception —while slices 0, 1, and 3 to 7 in a standard partition map are usually set up as a sequence of consecutive partitions of flexible size, slice 2 is usually laid out a single partition that covers the entire disk and overlaps with all other disk slices. This allows the system administrator to either use several slices out of 0, 1, and 3 up to 7 (in which case slice 2 becomes unusable) or (as often done with second, third, etc. disks) define an unpartitioned disk by using slice 2 without having to alter the partition map (of course, in this case all other slices are unusable)[4].

With standard Solaris 2.x, disk slices cannot be extended over multiple disks. The slice size is limited by the size of the hard disk. However, optional software from Sun called DiskSuite (bundled with the software for bigger server systems) permits having disk slices that extend over several physical disk drives. This software also allows dynamically increasing the size of a file system when more disk space becomes available.

A Typical Solaris System Disk Layout

The disk layout on Solaris 2.x systems depends heavily on the number and the size of the available disks. The user is reasonably free in deciding which parts of the UNIX file system to put on separate slices and on which disk (see also "Installing Solaris," page 39).The entire UNIX file system can be placed on a single, big disk partition (not recommended) or the standard recommendations as provided by the Solaris installation software can be used. The Solaris installation program proposes a layout for the installed disks, but the user can still decide to modify the proposed layout. For instance, it is up to the user to decide whether `/usr/openwin` should be made part of the `/usr` partition or whether it should be a separate disk partition.

The disk slices on a typical Solaris installation include the following file systems:

- Root partition, "/" (typically on `/dev/rdsk/c0t3d0s0`)
- Swap space, combined with `/tmp` (typically on `/dev/rdsk/c0t3d0s1`)
- `/var` file system (typically on `/dev/rdsk/c0t3d0s3`)
- `/opt` file system (typically on `/dev/rdsk/c0t3d0s4`)
- `/usr` file system (typically on `/dev/rdsk/c0t3d0s5`)
- `/export/home` file system (typically on `/dev/rdsk/c0t3d0s6`, sometimes also on a separate disk, such as `/dev/rdsk/c0t2d0s2`)
- Other file systems for extra disks, such as `/data` (e.g., on `/dev/rdsk/c0t2d0s2`)

These file systems are discussed in "Solaris Directory Structure," page 109.

How is a Disk Organized Internally?

Virtually all hard disk drives used today consist of multiple disk platters, typically rotating at 5400 or 7200 rpm (some even faster models are now becoming available). The disk platters are used on both sides. There is a fork-like arm that carries read/write heads for all of the platters (two per platter), and all disk heads are moved in parallel or synchronously.

Different from the old analog audio records, the information is not stored in a single, long, spiral track, but in concentric circles. The circle that is covered by one disk head (on one surface) in one disk turn is called a track. Each track is split into multiple data fields called

[4] Before file systems are installed in the various slices, partitioning consists only of a partition map, which was part of the disk header information (disk label) in the first sectors of the disk.

sectors. Typically, the sector size is 512 bytes. To maximize information density and to distribute the information density more evenly over the disk (the outer tracks are much longer than the innermost ones), some Sun disks define more sectors on the outer tracks than on the inside of the disk.

All tracks that can be accessed simultaneously without moving the heads are called a cylinder[5]. As head movements are mechanical and cost time (seek time, in the order of several milliseconds), Solaris tries to write consecutive data into sectors and tracks within the same cylinder. Also, because head movements take longer if they occur between distant cylinders, the Berkeley (ufs) file system tries keeping the information from a single file within groups of adjacent cylinders: ufs disks are organized in cylinder groups. A cylinder group typically is 16 cylinders.

When reading a specific sector on a disk, there is an additional delay from the time it takes until that sector is at the position of the heads. This delay is called rotational latency. Modern drives reduce the rotational latency by using faster rotation speeds. One type of drive, called a zero latency drive, start reading a track immediately after positioning the heads, storing the other sectors of a track in an intermediate buffer. When several consecutive blocks are read from disk, this additional information can be retrieved from a fast RAM buffer.

Zero latency cannot be used for writing to disk. The data for the sector currently under the write head may not be available yet. Instead, Solaris 2.x writes such information into a "virtual file system" (cache file system, `cachefs`) and updates the disk at a later point in time. For the user, writing to the disk this way takes almost no time at all. Also, information that resides in the cache file system can also be retrieved very quickly.

ufs File System

A typical ufs disk layout (header and first disk slice) contains the following elements:

- *Block (sector) 0* – Contains the disk label (disk type and characteristics, such as number of bytes per sector, sectors per track, tracks per cylinder, sectors per cylinder, total number of cylinders, and accessible cylinders) and the partition map (VTOC, unallocated disk space, partition numbers, size in sectors, starting sector, read-only and/or "unmountability" flags, mount points). The contents of the partition map can be read (as `root` only), using, for example:
  ```
  prtvtoc /dev/rdsk/c0t3d0s0
  ```
- *Sectors 1 to 15 on a boot disk* – Contain the boot blocks, a small program that is loaded into RAM at boot time by the firmware (PROM Monitor) and which then loads the kernel into memory. Only a disk with boot blocks can be used as boot disk. The boot blocks are installed (on the first disk only) by the installation software. A second disk could be made bootable (e.g., as a backup boot disk) by manually installing boot blocks using the command `installboot` (see Sun documentation for details).
- *Sector 16* – Contains the first superblock, which includes the fundamental parameters of a file system, such as the number of data blocks, and the number of I-nodes. Because the information in the superblock is so important for any file system, backup superblocks are stored in the first sector of every cylinder group. All superblocks within a disk slice are identical. If the first superblock is corrupted, the file system can be reconstructed using one of the backup superblocks, for example:
  ```
  fsck -F ufs -o b=32 /dev/rdsk/c0t3d0s5
  ```
  By convention, the first backup superblock is always stored at sector 32 of every slice.

5. The first cylinder is usually the most innermost, i.e., slices re filled starting with the inner cylinders.

- *Sector following every superblock* – Contains a summary block or cylinder group map, containing information about free data blocks and fragmentation in the cylinder group.
- *Sectors following a summary block* – In each cylinder group, the summary block is followed by the I-node table: a database (linked list) of available I-nodes for that cylinder group. The size of the I-node table depends on the number of I-nodes.
- *Following sectors in each cylinder group* – Contain I-nodes and data blocks. The number of I-nodes per cylinder group is usually adjusted such that there is one I-node per 2 KB in data blocks (i.e., actual disk space). The number of data blocks in a slice determines the total disk space, the number of I-nodes determines the maximum number of files (plain files, directories, symbolic links). For performance reasons, the usable disk space is usually adjusted to 90% of the total disk space (files filling the last 10% would be too fragmented).

Detailed information on the role of I-nodes and the structure of a UNIX file in the context of I-nodes and data blocks is given in "Hard Links, Symbolic Links: the UNIX File System," page 124.

Superblocks, summary blocks, the I-node library and the I-nodes themselves all take up disk space. Typically, about 6% of the physical space on a disk are used for such file system overhead. In addition to that, Solaris normally only fills a file system to 90% (i.e., if a slice is 90% full, the `df` command reports "100% full"). Overall, as shown in Table 9, this permits using about 85% of the physical disk size for data storage[6].

Table 9. Physical Disk Size Compared to Usable Disk Space

Model	*Size*	*ufs disk space*	*Usable disk space*
535 MB	521640 KB	498389 KB (95.5%)	448550 KB (86.0%)
1.05 GB	1026144 KB	963662 KB (93.9%)	867296 KB (84.5%)
2.1 GB	2077080 KB	1952573 KB (94.0%)	1757316 KB (84.6%)

6.2 Solaris Directory Structure

The next sections discuss what files are stored on the various disk slices. Although for most users it is not necessary to know which file is on what partition, it certainly helps for the system administrator, who is responsible for making sure that disk partitions are not (maybe inadvertently) overfilled. This not only promotes data integrity, but it also helps optimizing performance by avoiding operating with nearly-full file systems.

To see which disk slice your working directory is in, use the `df` command:

```
> df -k .
Filesystem         kbytes  used   avail  capacity Mounted on
/dev/dsk/c0t3d0s6  411582  340552  29880    92%     /usr
```

[6] It is possible to alter the ufs file system parameters (e.g., using the `tunefs` command or using `newfs` to generate a file system). The available disk space can be increased by reducing the number of I-nodes (but this may limit the number of files that can be stored on a slice) or by reducing the `minfree` parameter, i.e., the percentage of disk space that is usually kept free for performance reasons (the default is 10%, less may slow down the file system when it gets full). In general, altering the file system parameters is not recommended because the gain in disk space is very limited and normally not worth the effort.

To find out on which disk slice any other directory resides, enter:

`df -k` ***`directory_name`***

Files in the Root Slice

The top of the UNIX file system is root. There is also a primary (ufs) file system mounted on /, typically called the root slice or root partition. When starting up, the boot software first reads the boot block on the first disk sectors, and then it mounts the root slice that contains the UNIX kernel and software needed to continue the boot process. The directories in the root slice are described below. Many directories seem to be part of the root slice, but they actually are mount points for other file systems (disk slices and remote file systems).

`/bin`	Symbolic link to `/usr/bin.`
`/dev`	Pointers to devices, the definitions of the available hardware, including disk slices (e.g., `/dev/dsk/c0t3d0s0`, `/dev/rdsk/c0t3d0s2`), I/O ports (e.g., `/dev/ttya`, `/dev/term/a`), the console (`/dev/console`), pseudo-I/O ports that are used for windows and remote logins (e.g., `/dev/ptyp0`), the screen frame buffer (`/dev/fb`), and so on. To maintain compatibility with older software, many of the SunOS definitions are still present as symbolic links to the corresponding Solaris 2.x definitions (e.g, the SunOS names for disk partitions, such as `/dev/sd0a` and `/dev/rsd1b.` A special entry in `/dev` is `/dev/null`, the “bit bucket”, i.e., a port for output that is to be discarded.
`/devices`	The software port names in `/dev` are meant for the user, i.e., they should be both easy to understand and easy to memorize. For the software, however, accessing a port requires considerable extra information, and some of this may even be hardware-specific (i.e., it may depend on the workstation model). The port information for all ports is stored in the file name or path inside `/devices`, and the entries in `/dev` are symbolic links into the corresponding entries in `/devices`. When adding devices (peripherals, expansion boards, etc.) to the workstation configuration, appropriate new entries in `/dev` and `/devices` are generated by the software when booting up with the reconfigure option, i.e., with the `boot -r` command.

Some selected examples:

```
console -> ../devices/pseudo/cn@0:console
dsk/c0t2d0s0 ->
   ../../devices/iommu@0,10000000/sbus@0,10001000/
   espdma@4,8400000/esp@4,8800000/sd@2,0:a
dsk/c0t2d0s1 ->
   ../../devices/iommu@0,10000000/sbus@0,10001000/
   espdma@4,8400000/esp@4,8800000/sd@2,0:b
```

```
hme -> ../devices/pseudo/clone@0:hme
ie -> ../devices/pseudo/clone@0:ie
le -> ../devices/pseudo/clone@0:le
log -> ../devices/pseudo/log@0:log
mem -> ../devices/pseudo/mm@0:mem
null -> ../devices/pseudo/mm@0:null
```

```
openprom -> ../devices/pseudo/openeepr@0:openprom
pts/0 -> ../../devices/pseudo/pts@0:0
pts/1 -> ../../devices/pseudo/pts@0:1
rdsk/c0t2d0s0 ->
   ../../devices/iommu@0,10000000/sbus@0,10001000/
   espdma@4,8400000/esp@4,8800000/sd@2,0:a,raw
rdsk/c0t2d0s1 ->
   ../../devices/iommu@0,10000000/sbus@0,10001000/
   espdma@4,8400000/esp@4,8800000/sd@2,0:b,raw
term/a -> ../../devices/obio/zs@0,100000:a
term/b -> ../../devices/obio/zs@0,100000:b
ttya -> term/a
ttyb -> term/b
```

In most cases, the user needs only deal with the simpler entries in `/dev`, but there are exceptions—certain applications reserve a port by changing its permissions and ownership. For example, after setting up a printer on a serial port, the corresponding entry in `/devices` is owned by the user (and the group) `lp` and can only be accessed by that user (the symbolic links in `/dev` all have permission `rwxrwxrwx`). Occasionally, a port may be blocked for no good reason. In this case, the system administrator must fix the permissions and ownership of the corresponding port entries in `/devices`, not in `/dev`.

`/etc` Important UNIX and network administration files, such as `passwd` and `hosts`, as well as the system start-up scripts in the various `rc` directories (see "Run Level 1—Single-User Mode," page 95). In early SunOS releases, `/etc` used to also contain the system commands required for system administration and start-up. These have been moved into other directories (mostly `/sbin` and `/usr/sbin`), and symbolic links have been placed in `/etc` instead. Important files are stored in subdirectories within `/etc`, such as `/etc/dfs` (file system sharing administration), `/etc/lp` (printer administration files), `/etc/mail` (mail administration files), and others.

`/export` Disk slices and file systems to be shared with other systems via NFS mounting. On most systems, `/export` contains a single directory, `/export/home`, which is the mount point for the `home` directory slice (see "Home Directory Slice, /export/home," page 115).

`/home` In SunOS 4.x, the `home` directory slice was mounted on `/home`. In Solaris 2.x, this was changed to `/export/home`, and `/home` was kept as a directory for automounted home directories. *Do not use* `/home` *for standard home directories.* This will not work properly (this would happen in the `root` partition) and will confuse the automounter software. Typically, to improve performance in the VNMR environment, `home` directories are local, not automounted.

`/kernel`	Different from SunOS 4.x, where the UNIX kernel (the part of the UNIX operating system always resident in memory) was a single file `/vmunix` that was often regenerated and modified by the user, the Solaris 2.x kernel is now dynamic and usually doesn't need to be modified for the addition of new device drivers. Such drivers can be added dynamically at any time, without even restarting UNIX. The contents of `/kernel` are used to build a kernel (rarely an issue). The actual kernel is stored as `/platform/<arch>/kernel/unix`.
`/lib`	Link to `/usr/lib` (directory with libraries and compilation files).
`/lost+found`	Directory that takes up eventual unreferenced I-nodes when fixing the root file system using the `fsck` command.
`/net`	Main automounting directory for file systems other than `home` directories. All accessible exported file systems on accessible remote hosts (as set by `/etc/hosts`) can be accessed as `/net/`*`hostname`*`/`*`filesystem`*. To check if a remote host has exported file systems and what the exported file systems are, enter `cd /net/`*`hostname`* `ls`
`/platform`	Contains the actual UNIX kernel in a subdirectory named after the system architecture (kernels for different architectures can be stored on the same system). On a SPARCstation, the kernel is typically `/platform/sun4m/kernel/unix`.
`/proc`	Pseudo-directory (doesn't take up any disk space) containing a list of active system processes (file names correspond to the process-IDs).
`/sbin`	Important commands required during startup (before `/usr` is accessible) and during system recovery. This includes commands such as `ifconfig`, `init`, `mount`, `mountall`, `sh`, `su`, `swapadd`, `sync`, `umount`, `umountall`, and `uname`, plus the shell scripts `rc0`, `rc1`, `rc2`, `rc3`, `rc5`, `rc6`, and `rcS` that execute system start-up and shutdown scripts in `/etc/rc*.d`. See also "Run Level 1—Single-User Mode," page 95.
`/vol`	Used by the volume manager (`vold`) for removable media, such as CD-ROMs and floppy disks.
`/tftpboot`	Installed on UNITY*INOVA* spectrometer hosts as part of VNMR. It contains the acquisition computer operating system (VxWorks) and the Magnet and Sample Regulation (MSR) board, which is uploaded to these CPUs after rebooting the spectrometer console.
`/xfn`	Used in connection with the Federated Naming Service (FNS, as proposed by X/Open) for enterprise-wide system administration (used on top of NIS+). Not used in a typical VNMR environment.

Typically, the root slice contains also empty directories as the mount points for ufs mounting of local disk slices (`/var`, `/tmp`, `/usr`, `/opt`, `/export/home`, `/data`, etc.) for removable media, such as the CD-ROM (`/cdrom`) and floppy disks (`/floppy`), and for NFS mounting of remote file systems (often containing the file server host name).

/tmp Slice

The `/tmp` file system serves two purposes: `/tmp` contains temporary files (buffers for editing and scrolling windows and the like) that *are cleared automatically with every bootup*. Also, unlike SunOS 4.x, `/tmp` contains the primary swap space. The user cannot see as a directory listing any swap files inside `/tmp`. Swap usage is only displayed in the output of the `swap -s` command.

/var Slice

The `/var` slice contains files that are modified by every user's activities, such as spooling, system logging, and mail. The most important subdirectories include:

`/var/adm`	UNIX system administration files, statistics files, messages, logs.
`/var/cron`	Log file for some `cron` activities (typically `uucp`-related)
`/var/log`	Recipient for system log files. Mostly the `syslog` file and older, backed up versions (`syslog.0`, `syslog.1`, etc.) of the same file, which contains a detailed log about `sendmail` activities.
`/var/lost+found`	Takes up eventual unreferenced I-nodes when fixing the `/var` file system using the `fsck` command.
`/var/lp`	Logs (mainly error logs) from the printing services (`lp`).
`/var/mail`	Mail data for every user.
`/var/sadm`	Logs about the system software installation and the installation of Solaris patches.
`/var/spool`	Temporary storage for spooled data during the data transfer, for example, to a printer (`/var/spool/lp`) or from a mail server (`/var/spool/mqueue`).
`/var/tmp`	Under CDE, the user typically doesn't have an open console window. Console messages are stored in temporary log files with unique names such as `/var/tmp/wsconAAAa0037E:0.0`. It is not unusual to find a large collection of such console log files, using up lots of disk space.

Insufficient Space in /var

Iinstallation of patch clusters requires extra disk space. If don't have at least 36 MB of free space in `/var` (`/var/sadm`), the following error message appears:

```
Insufficient space in /var/sadm/patch to save old files
```

You can handle this problem three ways:

- Use the `-B` option while invoking `patchadd`. This option directs `patchadd` to save the backout data to the user-specified file system.
- Generate additional disk space by deleting files that are not needed.
- Override the saving of the old files by using the `-d` (do not save) option when running `patchadd`.

Also, see the manual page for `patchadd`:

```
man patchadd
```

/opt Slice

The `/opt` disk slice is meant for optional software. The purpose is to separate this software from user data. Software such as Wabi (Sun emulation software for running

Microsoft Windows under X), Web browsers, and third-party compilers are typically installed in `/opt`. This directory is usually 10 to 60 MB, often more.

/usr File System

The `/usr` file system contains the bulk of the UNIX software: commands, manuals, run-time library files, and so on. Its size (usually 100 MB to 400 MB or more) mostly depends on what software selections are made during the Solaris installation. This slice is meant to be used solely by UNIX and should not hold other data. During the installation, it may be necessary to leave free space in `/usr` for the installation of additional software that might be loaded later (see also "Installing Solaris," page 39). The files in `/usr` can be regarded as read-only by the normal user because only `root` can modify them—an important safety feature.

The most important directories and files in `/usr` are the following:

`/usr/bin`	Main directory for 400+ UNIX commands.
`/usr/ccs`	Directory with two subdirectories (`/usr/ccs/bin` and `/usr/ccs/lib`) containing commands (such as `as`, `make`, `lex`, `yacc`, and many others) and library files used with compilers and related software. Even though Solaris 2.x does not include a C compiler, the contents of `/usr/ccs` (which are only installed as part of the developer configuration) are required for third-party C compilers, such as GNU C (as used by VNMR).
`/usr/dt`	Directory for the CDE (Common Desktop Environment) software.
`/usr/include`	Directory with (Solaris- and system-specific) `include` files for C or C++ source files (installed as part of developer configuration).
`/usr/lib`	Large directory with object libraries (archives), runtime libraries, templates, and default configuration files for various software.
`/usr /lost+found`	Directory that takes up eventual unreferenced I-nodes when fixing the `/usr` file system using the `fsck` command.
`/usr/openwin`	Directory for the OpenWindows software. Depending on the available disk sizes and the disk partitioning, this may be on a separate disk slice.
`/usr/sbin`	Directory with system administration commands that are used by the operating systems (i.e., commands and daemons launched automatically when booting up Solaris) or are used only by `root`.
`/usr/share`	Directory with data that can easily be shared between different Solaris systems, even if they have different architectures. Includes `/usr/share/lib` (terminal information database, keyboard layouts, time zone definitions, etc.) and `/usr/share/man`, the ASCII UNIX manuals (actually `nroff` source files).
`/usr/ucb`	Directory with BSD UNIX commands (from SunOS 4.x) that are either not part of SVR4 UNIX or behave differently from their SVR4 counterparts. These files are kept for compatibility reasons only—it is better not to use them under Solaris 2.x (eventually they may no longer be supplied). Related data is stored in `/usr/ucblib` (runtime and object libraries for compiling and running commands under the BSD compatibility package) and `/usr/ucbinclude` (`include` files for C programs that differ between BSD and SVR4).

For reasons of compatibility between SunOS 4.x and other UNIX flavors, `/usr` also contains a number of symbolic links pointing to new file system locations: `/usr/5bin` (`/usr/bin`), `/usr/man` (`/usr/share/man`), `/usr/preserve` (`/var/preserve`), `/usr/pub` (`/usr/share/lib/pub`) and others, plus the subdirectories `/usr/adm`, `/usr/mail`, `/usr/spool`, `/usr/tmp`, which have all been moved to `/var`.

Home Directory Slice, /export/home

In a standard installation, `/export/home` contains both the VNMR software directory, `/export/home/vnmr`, as well as the home directory for every user (except `root`), such as `/export/home/vnmr1`, but all these can be installed wherever it is convenient from an operational point-of-view (e.g., on a big disk, such as `/data`). The VNMR software and the home directories can also reside on different slices. When processing large amounts of data, or when acquiring or processing multidimensional spectra, you should preferably place the home directories in a large slice or disk, with plenty of free disk space.

VNMR Software Directory

It is often advantageous to store the VNMR software directory (50 to 190 MB, depending on the spectrometer type and on the installed software options) in a place where two or more versions can coexist. Because the VNMR software directory is accessed either through a symbolic link with `/vnmr` or through an environment variable `vnmrsystem` defined for every user, VNMR can be stored in directories indicating the software revision, for example, `/export/home/vnmr53b`, `/export/home/vnmr61a`.

If the environment variable `vnmrsystem` points to `/vnmr`, it is only necessary to change the symbolic link `/vnmr` to change between different VNMR releases, for example:

```
rm /vnmr
ln -s /export/home/vnmr61a /vnmr
```

The files and directories inside `/export/home/vnmr` are described in the VNMR manual *Getting Started.*

Home Directories for Users

Home directories for VNMR users can contain numerous files:

- Files for customization and setup of the personal user interface, such as the directories `ib_initdir` (setup files for the image browser), `app-defaults` (customization files for VNMR-related programs and possibly also other applications), and `.dt` (setup files for CDE).
- Any number of personal data files and subdirectories.
- Optionally a local `bin` directory (e.g., `/export/home/vnmr1/bin`) for personal executables (compiled programs, shell scripts).
- Local `vnmrsys` subdirectory (one for each user), containing local, personal VNMR files (macros, pulse sequences and related files, menus, experiments, global parameters, etc.).

The `vnmrsys` directory has several subdirectories that also exist in `/vnmr`, such as `help`, `maclib`, `manual`, `menulib`, `parlib`, `psglib`, `seqlib`, `shapelib`, `shims`, and `tablib` (plus some optional subdirectories, such as `psg`). The convention is that the contents of these directories take precedence over files with the same name in a subdirectory of `/vnmr`. This way, every user can customize the VNMR software without having to touch the files in `/vnmr`.

vnmrsys Directory

The `vnmrsys` directory also contains the user's VNMR workspaces, `exp1` to `exp9`, as well as the user's global parameters (`vnmrsys/global`). These workspaces and the global parameters are standard UNIX files and directory trees; however, for speed reasons (disk access is very slow compared to memory-based operations), VNMR keeps the global parameters and the current experiment in memory, rather than working with the disk files.

The current experiment on the disk is only updated when joining a different experiment, upon exiting VNMR or when calling the VNMR command `flush`. The global parameters are only updated upon exiting VNMR or when calling the VNMR command `flush`. *Therefore, you must therefore not alter* `vnmrsys/global` *while VNMR is running.* Upon terminating, VNMR writes over the modified disk file with the version that it kept in memory. *For the same reason, you must not touch any current experiment.*

Experiment Locking

Considerable confusion could result if VNMR is running in multiple copies (e.g., via remote X terminals) and if several of these copies would point to the same experiment. To avoid this, VNMR uses a special experiment locking scheme.

Whenever VNMR is running, it generates an ASCII file `vnmrsys/lock_#.primary` (where # is the number of the current experiment). Therefore, the file `lock_1.primary` indicates that `exp1` is locked[7]. This file contains three items:

- *Lock type* – An integer number (see Table 10) indicating the type of lock.

Table 10. Lock Types

#	*Lock type*	*Remarks*
1	acquisition	`Acqproc`/`Expproc` related
2	background	`vbg`, or `Vnmr -mback`
3	foreground	normal, interactive VNMR operation
4	automation	special locking for automation mode

- *Host name* – The name of the host system on which VNMR is running
- *Process-ID* – The ID of the process (Vnmr) that is locking the experiment.

When joining a new experiment (through `jexp(#)`, `jexp#`, or upon starting up), VNMR first checks for the existence of a lock file for the target experiment. If such a file is found, VNMR then verifies whether on the indicated host (i.e., locally) there is a process with the process-ID found in the lock file. If the process exists, VNMR refuses to join that experiment. If the process is not found, that probably indicates that the copy of VNMR that generated the lock file somehow aborted or was terminated irregularly, such that it was not able to remove the lock file (free up the experiment) as normally when terminating via `exit` or when leaving an experiment via `jexp(#)` or `jexp#`.

Normally, the lock file is created and deleted automatically. Eventually, such a file can be left in `vnmrsys` by mistake (perhaps due to a software or computer crash). In most cases, the VNMR command `unlock(n)` will remove it. If the lock happens to be an acquisition

[7] In the process of establishing experiment locking, a secondary lock file is generated temporarily.

lock and is present by mistake, the lock file belongs to `root` (which runs `Expproc` or `Acqproc`), and `unlock` refuses to remove the lock file. However, the user can still remove that lock file using the `rm` command[8].

Files on Other Disks or Disk Slices

Systems with two or more disks usually have extra disk slices for data storage (the name `/data` is suggested). On systems with a small first disk and a big second disk, `/export/home` should be installed on the second disk, to allow for more flexibility with big experiments. The remainder of the first disk can then be mounted as `/data`.

Home directories can also be located in directories other than `/export/home`. Certain parts of VNMR (or customized macros) may assume that all home directories are in `/export/home`; therefore, a file in `/data` would not be found. This problem is bypassed by creating a symbolic link `/export/home/xx` pointing to `/data/xx`, such that for the software the file still seems to be in `/export/home`. Use the Solaris `admintool` to create new users. This way, it is assured that the password file `/etc/passwd` contains the proper information about the location of the home directory.

6.3 Multiuser Setup for VNMR

The directory `/export/home/vnmr` contains all the files that are (and must be) globally available. Because `/export/home` might contain various VNMR directories (e.g., `vnmr53b`, `vnmr61a`), to simplify access to the VNMR software directory, a symbolic link `/vnmr` is established, pointing to the currently active software directory. Usually, when addressing `/vnmr`, we actually don't mean the symbolic link, but the directory that link points to.

All users can read the data within `/vnmr` and also execute any macros, pulse sequences, shell scripts, etc. within this directory. In the default setup, only the NMR system administrator `vnmr1` can make modifications to files in `/vnmr`.

Every user has and can use their own (local) pulse sequences, macros, menus, parameter sets, shell scripts, etc. Each user can also modify files from `/vnmr` by creating a local copy and modifying that file (which would then be located in his own `psglib`, `maclib`, `menulib`, `seqlib`, `parlib`—all in the user's `vnmrsys` directory or within a local `bin` file for shell scripts).

If the user calls a macro that exists under the same name both in his local `maclib` and in the global `maclib` (`/vnmr/maclib`), the system executes the local macro, i.e., the local directories are looked up first. This is useful feature, but it can potentially create problems because the system always uses a local pulse sequence, while the global file (with the same name) may have been improved or upgraded.

The general idea is to keep the system software clean and to modify it as little as possible. If `vnmr1` wants to make changes available to all users, he or she may decide to put something into `/vnmr` on a case-to-case basis (make sure you keep a log of what modifications are done in `/vnmr`—this will be helpful when upgrading to the next VNMR release).

For macros, an additional directory can be searched automatically if this directory is specified in a (global) VNMR parameter `maclibpath`. This additional directory would

[8] The ability to remove a plain text file is not controlled by the permissions and the ownership of the file itself, but rather by the permissions of the directory in which the file resides.

be searched between the local and the VNMR `maclib`. This allows other users to also access the `maclib` directory of `vnmr1`, for example.

Chapter 7. Files, Permissions, and Owners

Sections in this chapter:

- 7.1 “How to Get File Information,” this page
- 7.2 “Protection Bits,” page 119
- 7.3 “Owners and Groups,” page 121
- 7.4 “Special Permissions,” page 122
- 7.5 “Hard Links, Symbolic Links: the UNIX File System,” page 124

An important activity in UNIX is file handling. This chapter covers how to get information on files, how files are protected from unauthorized use, and how links are used to associate files with each other

7.1 How to Get File Information

The UNIX commands `ls -l` or `ls -lF` can be used for getting a rather complete set of information on a directory or a files in a directory. For example, by entering

```
ls –ldF /dev
```

the information on the directory `/dev` might appear like this:

```
drwxr-xr-x  1  root   wheel    512   Dec 14  9:30  dev/
```

This displays the following information, from left to right:

- Type of file: `d` for a directory, `-` for a plain file, `l` for a symbolic link, `c` for a character special device (usually in `/dev`), and `b` for a block special device (usually in `/dev`).
- Permissions, or protection bits (the next nine characters, `rwxr-xr-x`).
- Number of hard links to the file (`1`).
- Name of the user who owns the file (`root`).
- Name of the group that also owns the file (`wheel`).
- Length of the file, in bytes (`512`).
- Date and time of the last modification (`Dec 14  9:30`).
- File name (with file type indication, if the -F option was used, `dev/`).

Each item is described further in the following sections.

7.2 Protection Bits

The protection bits for files are organized in three groups of three characters, `rwxrwxrwx`, where `r` is read permission, `w` is write permission, and `x` is permission to execute the file.

The first group (`rwx`) refers to permissions available to the user listed as the owner, the second group to the users in the group listed, and the third group to all users.

For directory listings, the meaning of the protection bits is slightly different: `r` is still read permission (enables listing the contents of the directory); `w` is the permission to write something into a directory (create a subfile), change it (rename a subfile), or delete contents (subfiles) of a directory; and `x` is the permission to make it the working directory (change directory into it). It is possible to have directories that a user can change into, but not read, and vice versa.

The permission flags can be set individually or all at once. Setting them all together is usually easier. The command to set all flags at once has the syntax

```
chmod number filename
```

where *`number`* is a three-digit number. Each digit has a value from 0 to 7 based on the sum of three permission bits, `rwx`:

- Read permission (`r`) adds 4 to a digit.
- Write permission (w) adds `2` to a digit.
- Execute permission (`x`) adds `1` to a digit.

For example, `rwx` is 7(4 + 2 + 1), `rw-` is 6 (4 + 2), and `r-x` is 5 (4 + 1).

For all three digits, add up decimal values as follows:

`400`	read permission for the user who owns the file
`200`	write permission for the user who owns the file
`100`	execute permission for the user who owns the file
`40`	read permission for users who belong to the group
`20`	write permission for users who belong to the group
`10`	execute permission for users who belong to the group
`4`	read permission for other users
`2`	write permission for other users
`1`	execute permission for other users

For example, to change the permission flags for the file `bin/shell_script` to `rwxrwxr-x`, which means no write permission for users outside the group, use the table to find that `rwxrwxr-x` is 775 (400 + 200 + 100 + 40 + 20 + 10 + 4 + 1), and then enter:

```
chmod 775 bin/shell_script
```

The `chmod` command has another method for setting individual protection bits. `chmod` can use the notation `u` for the owner, `g` for a group, and `o` for others, and then uses a plus (+) or minus (–) sign to either add (`+`) or subtract (`-`) `u`, `g`, and `o` with `r` for read, `w` for write, or `x` for execute. If any option isn't specified, all three groups are changed. The following examples should make this clear:

`chmod g+x file`	Adds permission for users who belong to the group to execute the specified file or directory.
`chmod +w file`	Adds permission for all three groups to write to the specified file or directory.
`chmod go-rw file`	Removes permissions for the users who belong to the group and all other users to read from and write to the specified file or directory.

There is also a recursive option:

`chmod -R 755 file`	Set `rwxr-xr-x` permission for the specified file or directory and all of their subfiles and subdirectories.
`chmod -R o+r file`	Adds read permission for the other users to the specified file or directory and all of their subfiles and subdirectories.

Setting permissions in absolute mode (e.g., `chmod -R 755 file`*)* in directories other than the personal data files and libraries can be dangerous, especially with the recursive option. When recursively changing permissions, it is better to selectively add or subtract permissions: the permissions for directories are usually not the same as for plain files, and this precludes combining the `-R` option with the absolute mode.

The permissions within the `root` and `/usr` partitions have been set up to ensure UNIX system security. It is not advisable to change permissions in this area because you may inadvertently erase special permission bits (see below) and eventually make UNIX non-functional and require reloading all the software.

7.3 Owners and Groups

Every UNIX file and directory has an owner assigned to it (as well as the dates of creation and of the last modification). Normally, the owner is the user that has created the file. Only the UNIX system administrator `root` can change the ownership.

Changing File Ownership

The command to change ownership is `chown`, which has the syntax
`chown` *`new_owner filename`*

This command has a recursive option (`-R`) for changing the ownership in all files and subfiles within a directory, for example, `chown -R vnmr1 *`

Although the `ls -l` command displays the owner's user name, the directory only contains the user's ID-number. Most other commands (like `tar` and `ps` with the appropriate options) only list this ID number. Instead of the owner's name, the `chown` command can also take the user's ID-number as an argument.

When `root` is loading the VNMR software, `chown` is used to make `vnmr1` the owner of most files within `/export/home/vnmr`; however, a special problem arises when `root` copies files into some user's home directory. Because `root` has created (copied) the file, `root` still owns it. Therefore, the user may not have permission to use it and, because the file belongs to `root`, the user cannot change the ownership. Whenever `root` copies files into someone else's home directory, `chown` should be used to also transfer the ownership.

The better solution would be if the user reads the file by copying it from outside into his or her own home directory because then, as the creator of the copy, the user becomes the owner of the file. Of course, this only works if the user has read permission for the file.

Changing Group Ownership

Similar mechanisms are used in connection with the group identification. All users are organized in groups (listed in `/etc/groups`) and every user also has a primary group ID-number (defined in `/etc/passwd`). Apart from that primary group membership, any owner can share another group's permissions, in that he can be listed as member of any number of other groups in `/etc/group` (of course, by opening up group memberships you are compromising in the area of security).

Users that have been defined with the command `/vnmr/bin/makeuser` are primary members of the group `nmr`, and they are all also members of the group `staff`, which is a standard group on most UNIX systems.

The same as the user ID, the group ID-number of a file can be manipulated. The `chgrp` command to change group ownership works exactly like the `chown` command (`chgrp` also has a recursive option). You must either be `root` or be owner of the file and member of the specified group to execute `chgrp`. As with `chown`, you can also specify the decimal ID-number instead of group name.

It is also possible to change or set the user and the group-IDs at the same time, using `chown` with the following syntax:

```
chown username:groupname filenames
```

Of course, also numeric IDs can be specified this way:

```
chown userID:groupID filenames
```

This syntax differs from SunOS and BSD UNIX, where a point is used instead of the colon.

7.4 Special Permissions

The permissions field can hold three more bits: the sticky bit, the set-UID bit, and the set-GID bit. These bits can be set with the `chmod` command similar to the other bits. Their decimal values are as follows:

4000	set-UID bit
2000	set-GID bit
1000	sticky bit

How does `ls -l` display these bits?

- If the sticky bit is set, the last `x` (execute permission for others) is replaced by a `t`, or to `T` if that last bit is not set (e.g., `rwxrwxrwt`).
- The set-UID bit changes the first `x` into `s`, or to `S` if that `x` is not set (e.g., `rwsrwxrwx`).
- The set-GID bit affects the second `x` the same as the set-UID bit.

The sticky bit is used in connection with executable programs. A program with the sticky bit is *not* swapped out, which can improve the performance of that specific program on systems with limited memory. On the other hand, because this forces swapping to occur with other programs, the performance of those programs may become worse. Because this may affect other users also, only `root` is allowed to set the sticky bit. The sticky bit is rarely if at all used today—it was created for systems with a small memory (64 MB or less) in the early days of UNIX.

If the set-UID bit is set on an executable file (a compiled program or a shell script), *whoever calls that program automatically bears the user-ID of the owner of the file, while that program is running* (and only inside that program and its child processes).

The set-GID bit works in an analogous way for the group ID-number. This feature is used a lot internally in UNIX, such as functions related to `login` (the login process is owned by `root`, after logging in all child processes are owned by the user that has logged in), and in fact many system functions need to change UID or GID.

It is quite obvious that both the set-UID and the set-GID bits are potential security holes. Therefore, these features should only be used where absolutely necessary. Special attention should be paid to programs (like shell scripts) that call child processes and may, in the case of problems, end up in an interactive shell. This may proliferate undesired permissions throughout the system. The most basic safety precaution for such files is, to *make them writable only by the owner*, particularly if they are shell scripts.

A typical example for the application of the set-UID feature is in the handling of optical media, where users should be able to mount and unmount disks but by default these are operations that can only be performed by `root`. Software for handling such removable disks (under SunOS 4.x also floppy disks) is based on shell scripts or programs that has the set-UID permission bit set. In earlier VNMR releases, shell scripts with set-UID permission were one of the possibilities to kill and restart `Acqproc` (which must be run by `root`, because it must be able to update FID files in each user's local directories).

In Solaris 2.x, removable media are automatically mounted via the `vold` volume daemon and handling the `Acqproc` process family is achieved via a special account `acqproc` (with or without a password) that has the same UID as `root` but, unlike a standard user account, does not open an interactive shell script after a login (or `su acqproc`). It only executes the shell script `/vnmr/bin/execkillacqproc`, which is owned by `root` and has the permissions `r-x------` (read and execute permission for `root` only) after that special account has been installed using `/vnmr/bin/makesuacqproc`.

We do not expect that you will ever need to set up software that uses the set-UID or set-GID protection bits; however, you should be abler to recognize files with these permissions, because they can be a serious threat to the security of your system. We will not discuss this in more detail here, but here is a list of files from `/usr/bin` and `/sbin` the use the set-UID permission bit[1]:

```
/sbin:
-r-sr-xr-x  1 root    sys      339780 May  3  1997 su

/usr/bin:
-r-s--x--x  1 root    sys      329360 Apr 26  1997 admintool
-rwsr-xr-x  1 root    sys       33260 May  3  1997 at
-rwsr-xr-x  1 root    sys       12080 May  3  1997 atq
-rwsr-xr-x  1 root    sys       10764 May  3  1997 atrm
-r-sr-xr-x  1 root    sys       19612 May  3  1997 chkey
-r-sr-xr-x  1 root    bin       14604 May  3  1997 crontab
---s--x--x  1 root    uucp      70664 May  3  1997 ct
---s--x--x  1 uucp    uucp      84524 May  3  1997 cu
-r-sr-xr-x  1 root    bin        9676 May  3  1997 eject
-r-sr-xr-x  1 root    bin       26312 May  3  1997 fdformat
-r-sr-xr-x  1 root    bin       28800 May  3  1997 login
-rwsr-xr-x  1 root    sys        9860 May  3  1997 newgrp
-r-sr-sr-x  3 root    sys       15688 May  3  1997 nispasswd
-r-sr-sr-x  3 root    sys       15688 May  3  1997 passwd
-r-sr-xr-x  1 root    sys       23740 May  3  1997 ps
```

[1] If there are substantial differences to how the permission bits are set for these files on your system, you are probably running an earlier version of Solaris 2.x and you probably have not have installed a security patch. Check the documentation that came with the Solaris software for patch information. To find out what patches have been installed on your system, use the command `showrev -p`.

```
-r-sr-xr-x  1 root    bin       18632 May  3  1997 rcp
-r-sr-xr-x  1 root    bin       64748 May  3  1997 rdist
-r-sr-xr-x  1 root    bin       14636 May  3  1997 rlogin
-r-sr-xr-x  1 root    bin        7940 May  3  1997 rsh
-r-sr-xr-x  1 root    sys       15820 May  3  1997 su
-rws--x--x  1 uucp    bin       56500 May  3  1997 tip
-r-sr-xr-x  2 root    bin       11192 May  3  1997 uptime
---s--x--x  1 uucp    uucp      66376 May  3  1997 uucp
---s--x--x  1 uucp    uucp      21428 May  3  1997 uuglist
---s--x--x  1 uucp    uucp      17772 May  3  1997 uuname
---s--x--x  1 uucp    uucp      62096 May  3  1997 uustat
---s--x--x  1 uucp    uucp      70268 May  3  1997 uux
-r-sr-xr-x  1 root    bin        4888 May  3  1997 volcheck
-r-sr-xr-x  2 root    bin       11192 May  3  1997 w
-r-sr-sr-x  3 root    sys       15688 May  3  1997 yppasswd
```

If you intend to use set-UID permissions on your system, check first whether there aren't any other solutions that are less dangerous.

7.5 Hard Links, Symbolic Links: the UNIX File System

In a simplistic view, the UNIX file system looks like a very large tree structure, but it has a number of very confusing aspects, such as different forms of links that may sometimes not be properly hierarchical and file systems mounted from various disk partitions, often including disk partitions from an other computer that are mounted locally via NFS.

Mostly, the user doesn't really care about the more complex aspects, because he or she is mostly concerned with the local file system subtree. Many users rarely look at files outside their own domain. Occasionally, the user stumbles over certain commands that sometimes refuse to work, such as when trying to move a directory (sub-)tree from one partition to an other one using the command `mv`. This section should help clarifying some of these issues.

In reality, the UNIX file system is even magnitudes more complicated than it seems from a user's point of view.

File System Tree

What is the structure of a UNIX file system tree? Most commonly, file system trees are visualized like Figure 16 (the lines are hard links, dotted lines are symbolic links).

This type of file system visualization—although simple and easy-to-read—makes it difficult to understand the difference between hard and symbolic links, and it is impossible to explain why you can have several hard links to a single file (with different names, even). Also, this scheme seems to suggest that files with real names exist on the disk, which is definitely not the case. In reality, files on a disk have no name, but they are stored under a number (the I-node number). For directories, you can assume that they all have the name "/" or "**.**" (they are also stored under an I-node number, as we will see below).

File Storage

Directories, plain files (text files, data files, programs), and even symbolic links are stored the same way— as a tree structure of their own (see Figure 17). At the top of the file tree is the so-called I-node (information node) containing:

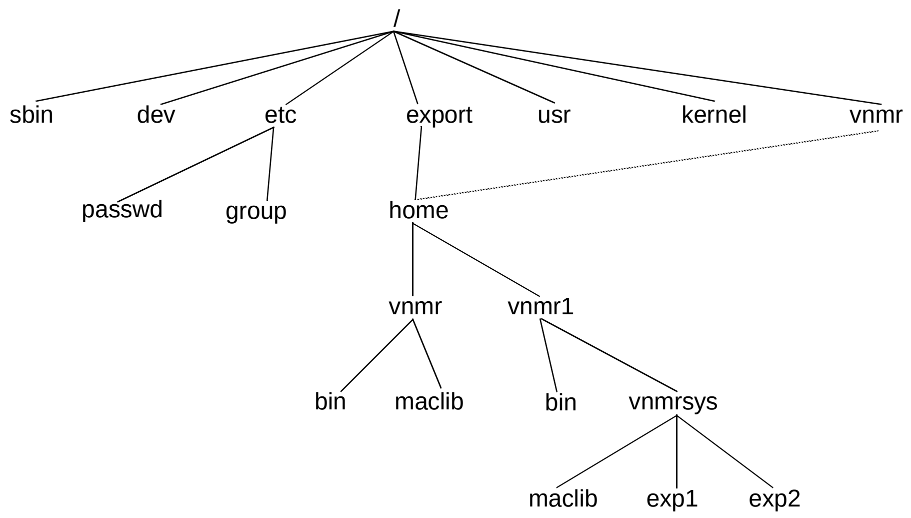


Figure 16. UNIX File System Tree

- 128 bytes of information that includes the protection bits, the number of hard links to the file, user- and group-IDs, and creation and modification dates.
- 12 addresses to 8-KB data blocks (on the disk, 16 adjacent sectors of 512 bytes are connected to a single data block).
- An address of an 8-KB data block containing 2048 addresses to 8-KB data blocks (the indirect blocks).
- An address of an 8-KB data block containing 2048 addresses to 8-KB data blocks, each containing 2048 addresses to 8-KB data blocks (double indirect blocks).
- An address of an 8-KB data block containing 2048 addresses to 8-KB data blocks, each containing 2048 addresses to 8-KB data blocks, each containing 2048 addresses to 8-KB data blocks (triple indirect blocks).

This results in a maximum size of 64 terabytes (TB) for a single file (70.4×10^{12} bytes).

Of course, real files only use rudimentary parts of this structure, and small files up to 95 KB do not even use the indirect block (the double-indirect block is only used for files larger than 16 MB, and triple indirect blocks would only be used for files greater than 32 GB). To further improve the disk space usage, UNIX stores small files and remainders of big files in so-called fragments (blocks of 512, 1024, 2048 or 4096 bytes), such that the smallest file consists of an I-node (128 bytes) plus a single block (fragment) of up to 512 bytes of data.

One danger with such a file structure is fragmentation. When a disk partition gets full, data blocks are collected from all over the disk. It is for good reason that UNIX tries to keep 10 percent of each partition free. To further improve the disk performance, Berkeley UNIX organizes the disk partitions in cylinder groups and tries to keep each file within a single cylinder group. Each of these cylinder groups includes a [backup]-superblock, a summary block, and an I-node table for disk administration (to keep track of free and used I-nodes, data blocks, and data block fragments). The data blocks and I-nodes are numbered within each partition—this implies that a file cannot extend over several disk partitions[2].

[2]. There is a special software from Sun, Solstice DiskSuite, that permits combining several physical disk partitions to a single (logical) partition. This software is normally only included with larger servers or with large disk arrays, but it can be purchased for workstations, too.

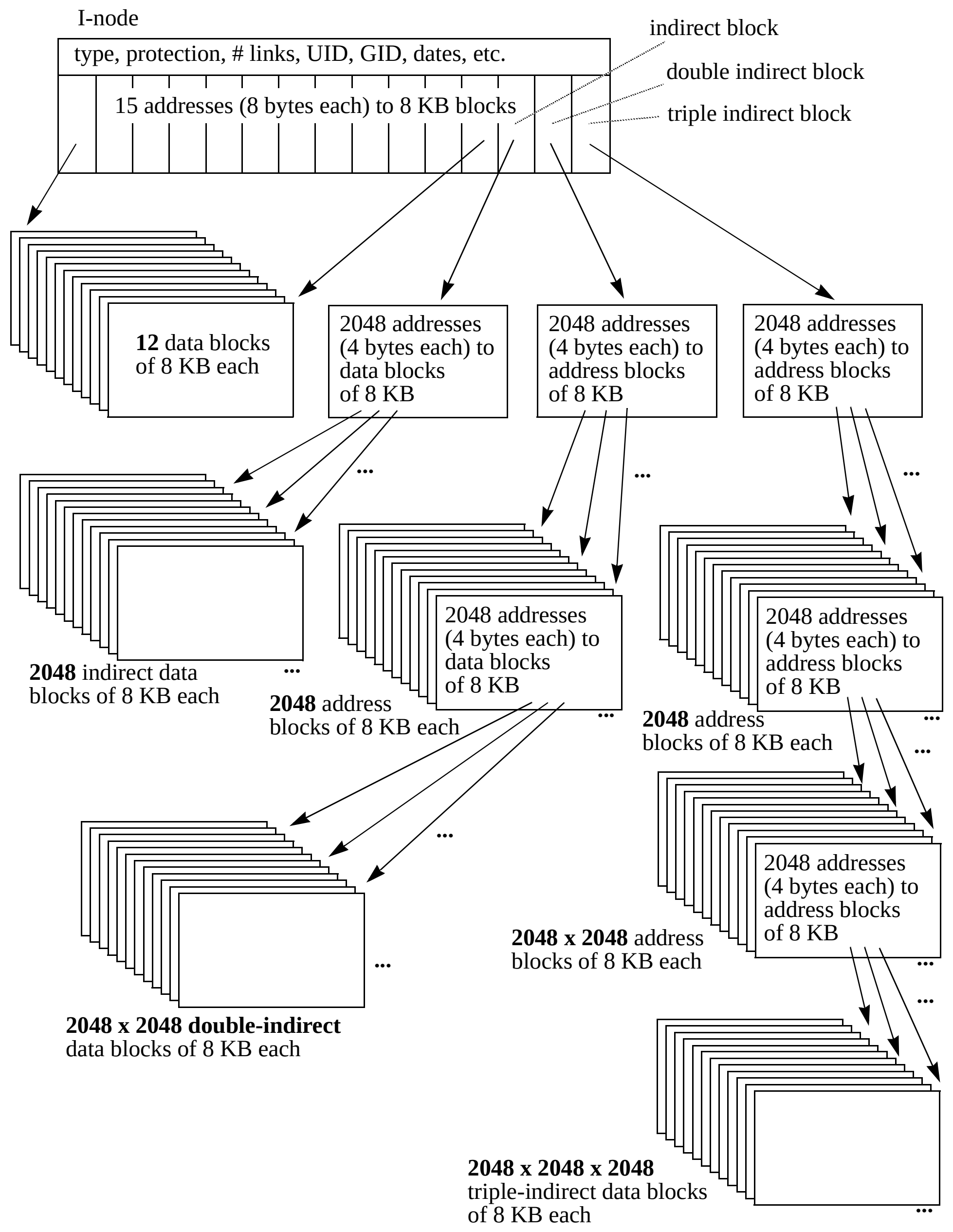

I-node
type, protection, # links, UID, GID, dates, etc.
15 addresses (8 bytes each) to 8 KB blocks
indirect block
double indirect block
triple indirect block
12 data blocks of 8 KB each
2048 addresses (4 bytes each) to data blocks of 8 KB
2048 addresses (4 bytes each) to address blocks of 8 KB
2048 addresses (4 bytes each) to address blocks of 8 KB
2048 indirect data blocks of 8 KB each
2048 addresses (4 bytes each) to data blocks of 8 KB
2048 address blocks of 8 KB each
2048 addresses (4 bytes each) to address blocks of 8 KB
2048 address blocks of 8 KB each
2048 x 2048 double-indirect data blocks of 8 KB each
2048 addresses (4 bytes each) to address blocks of 8 KB
2048 x 2048 address blocks of 8 KB each
2048 x 2048 x 2048 triple-indirect data blocks of 8 KB each

Figure 17. UNIX File Storage

Directory Files

A directory is nothing but a special type of plain file, consisting of an I-node and a variable number of data blocks or fragments. There are two kinds of information in a directory file: the I-node numbers (sometimes called the I-numbers) and the associated file names (variable length, up to 255 characters each). Such a directory entry (an I-node number and a file name) is called a hard link. It links a file to a directory. As mentioned above, I-nodes are numbered separately within each logical disk partition; therefore, hard links cannot point to files in a different partition (this can only be done through symbolic links, see “Symbolic Links,” page 128).

Hard Links

It now becomes obvious why and how a file can have multiple hard links to it—two directory entries can point to the same I-node. Hard links can only be made to files (I-nodes) on the same partition. The command to create *additional* hard links is `ln` (without the `-s` option), and a listing of the proper contents of a directory (I-numbers and file names) can be obtained using the `ls` command as follows:

```
# ls -ai /
    2 .                6444 devices         243 part
    2 ..               6339 etc            6413 pcfs
  258 .Xauthority      6336 export         6445 platform
 4414 .dt              6490 floppy            2 proc
  199 .dtprofile          3 home             22 sbin
  244 .fm              4242 kernel            2 tmp
 4436 .wastebasket       21 lib               2 usr
  231 TT_DB               3 lost+found        2 var
    4 bin              2128 mnt             250 vnmr
 4408 cdrom               2 net               2 vol
 2115 dev                 2 opt               4 xfn
```

Additional hard links can only be made to plain files, not to directories. A link to a directory could lead to cycles in the file system structure (if such a hard link points to a parent directory). Hard links to a file can be made from anywhere within a partition, but, because that can obscure the tree structure, it is *not a generally recommended practice*.

Additional hard links can be deleted (using `rm`) without deleting the file. A file is only really destroyed when the last hard link is removed. Such extra hard links are occasionally used as backup for safety purposes, preventing the loss of data if somebody by mistake erased an important file.

There is one place where (in earlier years) where VNMR used additional hard links. Sun-3 systems were available with two kinds of floating point math hardware—the 68881 math coprocessor and the Sun FPA (floating point accelerator). In order to optimize performance, programs that used floating point math extensively (`fitspec` and the `Vnmr` module itself) were compiled in two versions: one for each of the two versions of floating point math hardware. These modules were stored in `/vnmr/bin` under the names `fitspec_68881`, `fitspec_fpa`, `Vnmr_68881`, and `Vnmr_fpa`. During the installation, appropriate hard links named `Vnmr` and `fitspec` were made to these files (through `setfloat`), and a listing of these files on a machine with Sun-FPA could have been as follows:

```
# cd /vnmr/bin; ls -il fitspec* Vnmr*
2815  -rwxr-xr-x 2 vnmr1  1064960 Oct 12  1992 Vnmr
2814  -rwxr-xr-x 1 vnmr1  1040384 Oct 12  1992 Vnmr_68881
2815  -rwxr-xr-x 2 vnmr1  1064960 Oct 12  1992 Vnmr_fpa
2934  -rwxr-xr-x 2 vnmr1    40960 Oct 12  1992 fitspec
2933  -rwxr-xr-x 1 vnmr1    32768 Oct 12  1992 fitspec_68881
2934  -rwxr-xr-x 2 vnmr1    40960 Oct 12  1992 fitspec_fpa
```

Earlier VNMR releases used yet another link (named `master`) to the `Vnmr` module. Note that the number after the permissions (in the case of plain files) indicates the number of hard links to a file. Only the `-i` option to `ls` allows us to identify which hard links point to a particular I-node.

As shown in Figure 18, we can now visualize the “real” structure of directories and subfiles.

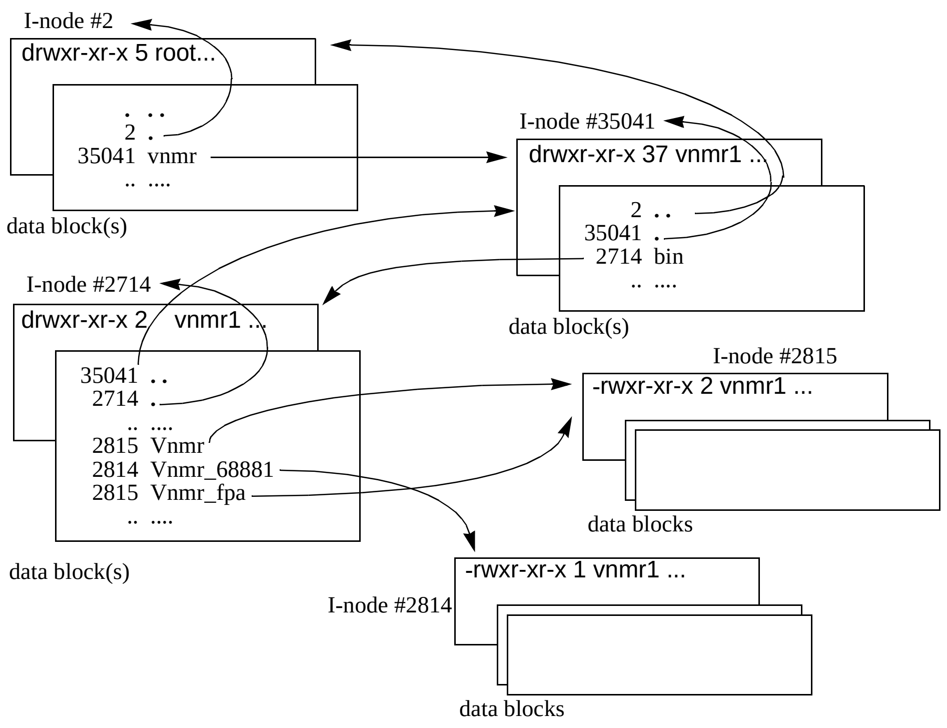


Figure 18. Structure of Directories and Subfiles

In VNMR, there are numerous cases of several macro entries in `/vnmr/maclib` or (even more so) multiple entries in `/vnmr/manual` pointing to the same data file. This is done using symbolic links (see below). In principle, it could also be achieved with extra hard links (that would save disk space), but the problem with this would be that commands such as `cp` or `tar` do not recognize that several hard links point to the same file. Therefore, after copying such a directory using `cp` or (preferably) `tar` (i.e., also after installing the software), you would end up with multiple copies of the same file.

You now can also understand the number of hard links in the output from `ls -l` in the case of a directory. There is one hard link from the parent directory, one hard link (“.”) from the directory itself, plus one hard link (“`..`”) from each of the subdirectories.

It was stated above that a single plain file cannot extend over multiple logical disk partitions. Data blocks are numbered within each logical partition. The same is true for the I-nodes. Therefore, hard links cannot be made to a different logical partition.

It is now obvious that it is not the files themselves that carry the file name but rather the hard links. This leads to a new type of tree structure (rectangles in the diagram are files, rounded rectangles are hard links), shown in Figure 19.

Symbolic Links

As shown in Figure 20 a symbolic link is a plain file with an I-node and a data block, but the data consist of a path name (relative or absolute, see “File Names,” page 133). Unlike a hard link, a symbolic link is an *indirect* connection to another file. Several other differences to hard links exist:

- Symbolic links can also be made between different partitions and disks.

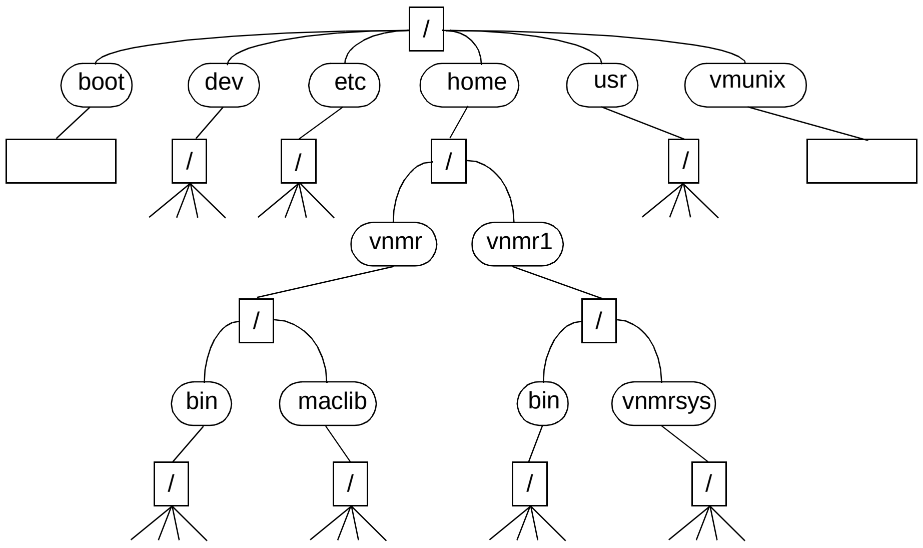


Figure 19. New Type of File Structure

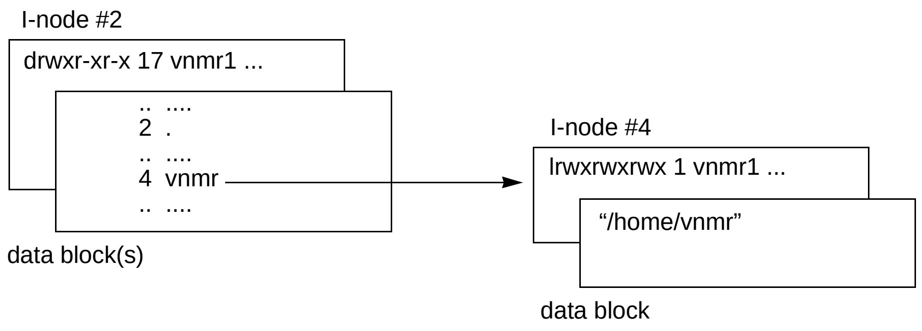


Figure 20. Symbolic Link

- Symbolic links are not verified upon creation. Only when trying to access a file through a symbolic link is the existence of the target file checked. Symbolic links can be deleted without affecting the target file. If the target file is deleted, the symbolic link remains unchanged.
- Symbolic links can also be made to directories.
- Symbolic links can also be made to parent directories.

Obviously, it is possible to create file system loops if a symbolic link is pointing to its parent directory or if two symbolic links are created that point to each other. In the case of links pointing to each other, the system would get trapped in an endless loop when trying to access that file. To avoid such cases, UNIX counts the number of symbolic links it passes, and it issues an error message "Too many levels of symbolic links" if it meets more than 20 symbolic links.

The case of a symbolic link pointing to its parent directory can have more serious consequences. The most important drawback of symbolic links is that they are not recognized as such by the `cp` command. The `cp` command copies the file a symbolic link points to, *not the symbolic link itself*.

Suppose a symbolic link inside `/export/home/vnmr` exists, pointing either to `/vnmr` (a symbolic link by itself) or to `/export/home/vnmr`. If `cp` is used to create a backup copy of the entire `vnmr` directory (`cd /export/home; cp -r vnmr vnmr.bk`), then `cp` follows the symbolic link and copies `vnmr` recursively into itself. It may ultimately fill the disk with copies of `vnmr` inside `vnmr` inside `vnmr`, etc.

Fortunately, there is an alternative for recursively copying directories containing symbolic links, involving the `tar` command (`tar` handles symbolic links properly, see "Using tar to Transfer Directories," page 153). These recommendations should help avoid problems:

- Avoid using `cp -r` whenever symbolic links could be involved.
- Use `tar` when copying directories containing symbolic links.
- *Avoid making symbolic links pointing to a parent directory.*

Chapter 8. UNIX Commands

Sections in this chapter:

- 8.1 “Command Syntax,” this page
- 8.2 “File Names,” page 133
- 8.3 “Important UNIX Commands,” page 135
- 8.4 “Interactive Work with the C Shell,” page 138
- 8.5 “Shortcuts within Open Look and CDE,” page 139

Modern software user interfaces make administration less and less dependent on UNIX knowledge. Under OpenWindows, for example, Admintool simplifies administration of users, groups, hosts, remote and local printers, serial ports, and UNIX software. File Manager, Text Editor, Print Tool, Tape Tool, and Mail Tool are also convenient mouse-driven OpenWindows utilities.

However, there are still a number of cases where the System Administrator needs to go to the UNIX operation level. This chapter describes UNIX command syntax and file names, and lists a selection of UNIX commands, many of which you will find of importance.

8.1 Command Syntax

UNIX command names can be any reasonable length. They typically contain a combination of alphanumeric characters, including special characters such as the underscore. UNIX is case-sensitive, which means it always distinguishes between uppercase and lowercase.

The UNIX command separator is the semicolon or the Return key. If several commands are entered on the same line, they must be separated by semicolons.

Command Arguments

The UNIX argument separator is the space. Arguments are separated from the command and from each other by spaces.

Argument types are numeric, names, strings, or options. Simple strings (no spaces or special characters) can be treated like names using the space as a delimiter; complex strings must be enclosed in double straight quotation marks (“...”). Characters within strings which belong to UNIX-shell commands like a single quotation mark must be escaped with the backslash character (\) (see UNIX manuals for additional details). Options are usually a selection of characters, preceded by a minus sign (`-`). Some commands, such as `tar`, have a special syntax, where the minus sign must *not* be used for the options.

Online Command Reference

Online help on UNIX command syntax is available with the `man` command. To display help on the `tar` command, for example, enter:

```
man tar
```

This only works if the manuals were installed onto the system. The manuals are stored in several megabytes of unformatted `nroff` files. The `nroff` command slows down the manual display, but manuals can be preformatted with the `catman` command. This takes a several minutes to process. After preformatting, the original `nroff` files can be deleted with `rm -r /usr/man/man*`. This speeds up the `man` command without using up much more disk space.

The `man` command does not know about aliases. Entering `man lf` results in an error message. Use `alias` to find out about all presently defined aliases in the current C shell.

Command Grouping

Commands can be executed in a separate shell if they are enclosed in parentheses (...). This has two consequences—their action does not affect the current shell and they behave like a single command for the calling shell. For example, if the directory should be changed for only a few commands, execution in a separate shell does not change the current working directory, for example:

```
(cd /vnmr; mkdir test)
```

If several commands inside parentheses produce output, this output is concatenated, for example:

```
(echo "file listing:"; ls -l) > listing
```

As long as a command is running, it runs exclusively in the shell in which it has been submitted, and standard output, as well as any error output, is made to the same shell.

Background Execution

Commands can also be executed in background. This means that the shell may accept and execute one or more of the next commands while the previous command is running. Background calls are made by attaching an ampersand (&) to the command, for example opening `mailtool` in background operation:

```
mailtool&
```

Input and Output Redirection

With any command that makes standard output, this output can be rerouted into a file:

`ls`	Prints the catalog of the current working directory
`ls >` *`listing`*	Writes the catalog into the text file *`listing`*
`ls >>` *`listing`*	Appends the catalog to the text file *`listing`*

If the file does not exist yet, it is created by the command. If the file does exist, the command `ls >` *`listing`* overwrites its contents.

Commands that expect keyboard input (called standard input) can take that input from a text file instead:

`wc -w <` *`listing`*	Counts the words in the text file *`listing`*.

Output from one command may be directly used as input for the next command. This UNIX feature is called a pipe and is represented by a vertical bar (|):

`ls -l	more`	Displays a long listing with the `more` command.

Multiple pipes can be chained together, for example:

```
ls -lR | grep root | more
```

Command substitution can be used to insert the output of a command into a text string:

```
set prompt="‘uname -n‘:‘who am i‘ \!>"
```

This inserts the output of the commands `uname` and `who am i` into the prompt string. Commands in back straight quotes (‘...‘) are executed first, and their output substitutes for the quoted strings before the command line is interpreted completely.

8.2 File Names

UNIX file names can be almost any length (up to 256 characters) and can contain both uppercase and lowercase characters, numbers, and some special characters (including the underscore, dot, hash sign, and plus sign).

File Name Suffix

Some commands and shell scripts expect file names with a specific suffix, such as `filename.c` for C programs), but generally a suffix has no function under UNIX. Therefore, users are free to make up own set of suffixes, perhaps using `filename.old` for annotating old versions of files, `filename.bk` for backup copies, and so on.

In fact, the suffix can be used as a standard file-name extension. VNMR uses the `filename.fid` suffix for FIDs, `filename.par` for parameters, and various others. Only one suffix per file name is recommended. If the file name is very long and needs some structuring, you can use underscores or uppercase characters.

Dot Files

A special class of files, called *dot files*, has a file name that starts with a period (e.g., `.login`). Dot files are not displayed in standard file listings using the command `ls`. To include dot files in the file listing display, enter `ls -a`. See Chapter 13, “UNIX System Customization,” for more information on system dot files.

File System Hierarchy

The slash (/) is used to separate directory levels. Under UNIX, a file name, whether simple or complex, is also called a path because the name describes a path to a specific file. There are two classes of paths:

- Absolute paths start at the root (/) and lead all the way down to the file, for example, `/export/home/vnmr1/vnmrsys/exp1/acqfil/fid.`
- Relative paths reference the current directory the user is in, for example, `vnmrsys/exp1/acqfil/fid`.

The current directory of the user is commonly called the *working directory*. The path to this directory can be displayed with the command `pwd`. The working directory also has a file name: the dot (.). Therefore, the relative paths `vnmrsys` and `./vnmrsys` are identical.

The directory that automatically becomes the working directory after login is called the *home directory*. From within a C shell (`csh`), a user can reference his or her home directory with a tilde "~". Another user's home directory can be referred to with the file name `~`*`username`*. Note that within a Bourne shell (see "Bourne Shell Scripts," page 212) these abbreviations are not valid.

The directory above the working directory is called the *parent*. This directory has two dots (`..`) as the file name (e.g., `cd ..` changes the working directory to the parent directory).

Wildcard Characters

Most UNIX commands accept wildcard characters in file names:

- An asterisk (`*`) stands for any number of characters.
- A question mark (`?`) stands for a single nonblank character.

By using square brackets `[...]`, restrictions for single character positions are also possible:

- `[a-z]` stands for any lowercase character.
- `[abc]` stands for either a or b or c.

These wildcards can also be combined, such as `??*` stands for "at least two characters". The following are examples of wildcard characters in use:

`rm *.o`	Removes all files with `.o` suffix
`ls a*`	Lists all file names starting with `a`
`rm -r par[2456]00`	Removes files `par200`, `par400`, `par500`, `par600`
`ls .??*`	Lists all dot files with at least 2 characters after the dot.

Keyboard Shortcuts for Entering File Names

If you are using the C-shell and the `filec` option is set by entering

```
set filec
```

(or by including `set filec` in your `~/.cshrc` file for permanent setting, undo it by the `unset` command), the Esc and Ctrl-d keys are redefined in certain situations to display file names in the path you are entering. For example, enter

```
ls /vnmr/n
```

and press the Esc key. Upon pressing Esc, the system completes the command line to

```
ls /vnmr/nuctables
```

You can now press Return to execute the line.

For another example, enter

```
ls /vnmr/a
```

and press Esc. The system fills in two characters

```
ls /vnmr/acq
```

and then beeps, because `/vnmr` contains several files that start with the character `a`—`acq` is the common part of all files starting with `a`. The beeping can be switched off with a line

```
set nobeep
```

in your `~/.cshrc` file. You could now continue by typing one or two more characters, and then press Esc again, and so on.

You can also reduce the number of files that the Esc option includes. For example, including

```
set fignore .o
```

in your `~/.cshrc` file causes it to disregard files with `.o` suffix.

Before or after using Esc (in fact anywhere within a command line), you can press Ctrl-d. The system displays a list of files that match the characters in the last token (such as `ls a*` or `ls acq*`) and then repeats the prompt and the previous (unfinished) command line, allowing you to make a selection. For example:

```
> ls /vnmr/user [Ctrl-d]
user_templates/ userlib/
> ls /vnmr/userlib
```

This is a special use of Ctrl-d. Under other circumstances, Ctrl-d works differently.

Note that with the `filec` option, Ctrl-d always has this functionality in a C shell. As soon as the first character on a command line has been typed, it interprets the last token on the line as a file name and tries to complete it. Therefore, it is impossible to kill a window or log out with Ctrl-d inside a command line. It may be necessary to make the system forget the partial command line by pressing Ctrl-u first.

8.3 Important UNIX Commands

Some of the more useful and important UNIX commands are listed in this section. Commands listed below in bold are important for standard UNIX users; the other commands are often found in shell scripts or mainly important for UNIX administrators. Refer to Sun manuals for complete information on UNIX commands.

Directory Commands

`pwd`	Displays working directory.
`cd`	Changes to home directory.
`cd` *`directory`*	Changes working directory to *`directory`*.
`ls`	Lists files in working directory.
`ls` *`directory`*	Lists files in *`directory`*.
`lf`	Alias of `ls -F` (defined in the file `.cshrc` distributed with VNMR) that is like `ls` but with file type annotation (`/` marks a directory, `@` a symbolic link, and `*` an executable file).
`ll`	Alias of `ls -l` (defined in the `.cshrc` file distributed with VNMR) that is like `ls` but with additional file details.
`mkdir` *`directory`*	Creates new directory with the name *`directory`*.
`rmdir` *`directory`*	Removes directory with name *`directory`*, provided *`directory`* is empty (i.e., has no files).

File Handling Commands

`cp` *`file1 file2`*	Copies file named *`file1`* to file named *`file2`*. If the name *`file2`* exists, it is overwritten with no warning message (unless the `-i` option is used). `cp` works only on simple files, not on directories.

`cp` `-r` *`file1 file2`*	Makes a recursive copy for a directory and all subfiles. It does not recognize symbolic links (use `tar` instead). Beware of recursively copying a directory into itself—this can quickly fill a file system, and the resulting infinitely stacked directories is difficult to clear.
`mv` *`file1 file2`*	Moves one or more files (*`file1`*) into another directory (if *`file2`* is a directory). If *`file1`* has new names, it also renames the files. If *`file2`* already exists and is not a directory, the file is lost without warning message (unless the `-i` option is used). Directories cannot be moved between different file systems (partitions); use `cp` instead.
`rm` *`files`*	Removes one or more files (*`files`*), accepts wildcards (be careful), but does not work for directories, which use `rmdir` or `rm -r` instead.
`rm` `-r` *`directory`*	Removes directory named *`directory`* recursively with all of its subfiles.
`tar`	Reads and writes files and directories (see Sun manuals and Chapter 10, "Using Tapes and Floppy Disks," for details). `tar` may also be used to copy directories recursively across a network.
`ln -s` *`file1 file2`*	Creates symbolic link for *`file1`* to destination path specified by *`file2`*.
`touch` *`files`*	Creates or updates directory entry for one or more files (*`files`*) If file does not exists, creates a file of size zero; if file exists, updates date of last modification.
`find`	Searches in a given directory, downward through all directory levels, trying to the find one or more files. As shown with the next few examples, `find` has various options: where to search, by what criterion to find files (for example, by name, owner, size, permission, access date, modification date, owner, and number of links), what to do with the files found, and what to do with the information (print it or execute some other command).
`find /home -name "*.old" -user vnmr1 -print`	Finds all files with `.old` suffix within directory `/home` that belong to user `vnmr1,` and print the paths to the files.
`find / -name core -exec rm {} \;`	Finds all files named `core` and removes them. Note the notation for the arguments to the `rm` command, and the backslash for the semicolon (the semicolon should not be interpreted by the shell as command separator, but as a terminator for the arguments of the `rm` command)
`find / -atime -3 -print`	Finds all files that have been accessed within less than 3 days and prints their paths.
`find / -mtime +30 -print`	Finds all files that have last been modified more than 30 days ago and prints their paths.
`chmod`	Change permissions (see Chapter 7, "Files, Permissions, and Owners," for further information).
`chown`	Change owner (see Chapter 7, "Files, Permissions, and Owners," for further information).

Text Commands

`cat`	Displays text from text files or writes it to standard output.
`head`	Displays text from the start to a specified point in a text file or from standard input.
`tail`	Displays text from a specified point to the end of a text file or from standard input.
`cut`	Cut out the specified character or token column(s) from a file or from standard input.
`fold`	Displays text from a text file or standard input, wrapping around (folding over) long lines.
`more`	Displays text from a text file or standard input in pages with the following keyboard commands: press Spacebar to scroll to the next page, Ctrl-b to scroll back one page, Return to scroll down one line, and q to quit. When `more` is used on a text file, `v` switches to the `vi` editor (after quitting `vi`, `more` becomes active again).
`page`	Similar to `more`, but erases and redraws screen (window). `page` is somewhat faster than `more` and uses the same keyboard commands as `more`.
`vi`	Edits a text file (see "Text Editor vi," page 142)
`textedit`	Edits a text file (see "Text Editors dtpad and textedit," page 141)
`sed`	Edits a stream of text.
`diff` *`file1 file2`*	Prints the differences between two files.
`diff -r` *`dir1 dir2`*	Recursively compares two directories and their subfiles.
`grep` *`pattern files`*	Searches for a pattern within one or more text files.
`grep -l` *`pattern files`*	Returns the names of files that contain the specified pattern.
`awk`	Scans and processes text file line by line.
`tr`	Translates characters, for example, converting upper case to lower case.
`fmt`	Format ASCII text: wrap lines in paragraphs, does not split words (unlike `fold`).
`sort`	Sorts lines in a text file.
`nroff`	Formats text for character-oriented terminals.
`troff`	Formats text for pixel-oriented terminals.
`echo`	Displays the text string that is given as an argument.

Process Management and System Administration Commands

`df` `-k`	Displays percentage of free disk space in each (or specified) partition.
`du` `-k`	Displays size in kilobytes of files in the current working directory and all subfiles.
`du` `-k` *`file`*	Displays size in kilobytes of all subfiles of file *`file`*, including their subfiles.
`du` `-sk` *`files`*	Displays total size in kilobytes of one or more files (*`files`*), including all its subfiles.
`du` `-sk`	Displays total size in kilobytes of the working directory.
`su`	Requests changing user-ID to become `root`. If passwords are implemented, `su` asks for the `root` password.

`su - user`	Request changing user-ID to `user`. The first argument [-] changes the environment to what would be expected if the user actually logged in as the specified user. If passwords are implemented, `su` asks for the `user` password.
`su root -c 'kill #'`	Kills a process with PID # as a temporary superuser. Asks for the superuser (`root`) password.
Ctrl-d	Logs user out. Exits the current window or subshell.
`ps -ef`	Displays all process-IDs and status of current processes.
`kill`	Sends signals to specified processes (requires process-IDs); the default signal is 15 (TERM) and causes most processes to terminate. If the TERM signal fails to kill a process, signal 9 (KILL) should always do the job: `kill -9 PID`.
`sync`	Updates (synchronize) the system disk for a consistent file system. The `update` daemon does a periodic `sync`.
`prtvtoc /dev/rdsk/#`	Displays information on format and partitions of disk #. To identify #, use the `df -k` command (see above).
`date`	Displays system date and time.
`who`	Displays a list of users currently logged in.
`who am i`	Displays current user name, the terminal, and the login time.
`hostname`	Displays the name of the current host.
`uname -n`	Displays the name of the current user.
`uname -a`	Displays all information about the current system.
`id -a`	Displays user and group ID.
`compress` *`file`*	Compresses a (plain) file.
`uncompress` *`file`*	Uncompresses a compressed file.

8.4 Interactive Work with the C Shell

Commands in bold are important for standard UNIX users; the other commands are often found in shell scripts or mainly important for UNIX administrators. Refer to Sun manuals for complete information on UNIX commands.

`alias`	Displays currently defined aliases.
`alias lf ls -F`	Makes the command `lf` an alias of `ls -F`.
`unalias lf`	Removes alias so that alias `lf` no longer valid.
`history`	Displays last command lines (recommended alias: `h`)
`!!`	Reexecutes last command line.
`!!`***`str`***	Reexecutes last command line with string *`str`* added.
`!`***`n`***	Reexecutes command line number *`n`*.
`!`***`-n`***	Reexecutes command line *`n`* lines back.
`!`***`str`***	Reexecutes command line that starts with *`str`*.
`!`**`$`**	Reexecutes last argument (token) from last command line.
`^`***`str1`***`^`***`str2`***	Replaces *`str1`* by *`str2`* in last command line and reexecutes.
Ctrl-c	Cancels execution of the foreground command.
Ctrl-s	Stops execution of the foreground command (no command entry possible, except after Ctrl-q or Ctrl-c).

Ctrl-q	Resumes stopped or suspended command.
Ctrl-z	Suspends execution of the foreground command (command entry possible).
`bg`	Resumes execution of suspended command in background.
`jobs`	Displays information on background jobs.
`fg`	Brings background process to foreground.

8.5 Shortcuts within Open Look and CDE

Window moving and resizing is accomplished in Open Look with the left mouse button.

- To move a window, grab the window frame at any point and drag it to the new position. If necessary, windows can pass the edge of the screen.
- To resize a window, grab one of the resize corners and move it

For transferring text within or into `vi`, only the Copy and Paste keys (L6 and L8, or the corresponding window menus) can be used.

Within a shell (`csh`) or when working with the Open Look version of `textedit`, you can use the powerful drag-and-drop utility (apart from the Cut, Copy and Paste keys on the keyboard):

1. Highlight the text to be transferred with the left or middle mouse button, and then move the mouse over the highlighted text and press the left mouse button. A small rectangle with an arrow in the upper-left corner appears
2. Move the cursor over to the position where you want to insert that text (when editing) or over the window with the interactive shell into which you want to drop the text.
3. Release the mouse button. The text is transferred.

Within `textedit`, when you release the button over a character, the text is inserted before that character

Drag-and-drop does not work from and into nonscrolling windows (i.e., also scrolling windows after disabling the scrolling) and from and into `vi`. This is a limitation of the current version of Open Look, not a bug. Drag-and-drop also does not work within a remote shell and from and into the VnmrX windows.

With the new UNIX windows environment, CDE (Common Desktop Environment), most of the restrictions in Open Look are overcome. It is a next step towards a windows-driven interface as it has been used on personal computers for many years. Because CDE has many similarities to the personal computers and is easy to explore by trial and error, no further descriptions are given here, except that to copy text within the same window in CDE (e.g.: into the command line), highlight the text that you want to copy, and then press the middle mouse button. This pastes the highlighted text at the current input location (e.g.: the command line).

Chapter 9. Text Editing

Sections in this chapter:

- 9.1 "Text Editors dtpad and textedit," this page
- 9.2 "Text Editor vi," page 142
- 9.3 "Other Editors," page 147

UNIX includes several programs for editing text, including `textedit`, `vi`, `ex`, and `ed`. Each is described in this chapter (`textedit` and the line editors `ex` an `ed` only briefly):

For further information, refer to Sun manuals. Refer also to the *VNMR Command and Parameter Referenc*e.

From VNMR, UNIX editors can be called directly, for example, by `macroedit` or `macrovi`. The VNMR command `macroedit` calls an text editor that is defined in the variable `vnmreditor`. It can be changed by

```
setenv vnmreditor new_editor
```

where *`new_editor`* is one of the editors listed above, and can be made active by recalling VNMR from this shell. For a permanent change, add this command to the user's `.login` file. The VNMR command `macrovi` always uses the `vi` editor.

9.1 Text Editors dtpad and textedit

The OpenWindows `textedit` program is a mouse-oriented, window-based text editing tool that always runs in its own window. Therefore, it cannot be used with windowless environments. `textedit` is called with the command `textedit` *`filename`* from the root menu or from a shell, like a shelltool window. Within CDE, a similar but more comfortable text editor, `dtpad`, can be called by mouse clicking on the editor icon.

The `textedit` program is easy to learn. It is always in the insert mode, the mouse can be used to set the cursor point to insert or delete characters, and a pop-up menu allows access to more complex functions.

Within the Open Look GUI, `textedit` supports the drag-and-drop feature (see "Shortcuts within Open Look and CDE," page 139), which makes this editor even more attractive. `textedit` can also be made to automatically wrap the lines at word breaks, not just at any character. This makes it look like it is automatically formatting lines. However, note that `textedit` in such a case does *not* insert a newline character.

When using `vi` for editing a file that has been generated or modified using `textedit`, you may encounter very long lines of text. With `dtpad`, the CDE text editor, this can be avoided: it is possible to have linefeeds inserted in wrapped lines when a text is saved.

You can access the Edit menu easily, which gives you fast and direct access to the Again, Undo, Copy, Paste and Cut functions. Within the Open Look environment, all menus (except for the Extras) are available from the top of the window as well as within the window (menu mouse button).

Experienced UNIX users will probably prefer the `vi` text editor, which is more complex (three modes) and has a less sophisticated user interface (the mouse is not used) but is faster and more powerful, especially for repetitive tasks such as making global changes. The real advantage of `textedit` and `dtpad` is that they can be used reasonably well after a few minutes of training, whereas `vi` requires much more time to learn completely.

9.2 Text Editor vi

The `vi` program (pronounced “vee-eye”) is the standard full-screen editor under UNIX, compatible with all full-screen terminals and RS-232 terminals. Although the `vi` editor is powerful, it is much more complex than `textedit and dtpad`, and the user interface for `vi` is not as sophisticated. `vi` does not use a mouse or menus and provides very little status information to the user. On the other hand, `vi` is particularly well suited for editing line-oriented documents such as source code, tables, and UNIX text definition files.

`vi` is called with the command `vi` *`filename`*. If `vi` is entered with no argument, the file to be edited can be read in from within the editor. If a single file is given as an argument, `vi` opens with that file on display. If multiple file names are given (wildcard characters can be used), `vi` reads each file in sequentially for editing one at a time.

`vi` operates in three modes, called command, insert, and last line.

Command Mode

`vi` starts in the *command mode*, where you can do the following:

- Move the cursor around the file
- Enter commands to work with the existing text
- Change to the insert mode to add text
- Change to the last line mode quit `vi`, read another file, etc.

A sample of commands in the command mode is given below. Because there is no command line, the commands do not show up on the screen but are executed immediately without pressing the Return key.

Moving Around the File

To move around in a file, press the up-down-left-right arrow keys on the keyboard or enter the following commands:

`k`	Move cursor up one line
`j`	Move cursor down one line
`l`	Move cursor one character right
`h`	Move cursor one character left
`G`	Go to the start of the last line in the file
`3G`	Go to the start of line 3
`0` (zero)	Go to the start of the current line
`$`	Go to the end of the current line
`w`	Go forward one word (to the start of the next word)
`3w`	Go forward three words

`b`	Go backward one word (to the start of the current word)
`5b`	Go backward five words
Return	Go to start of next line
`3` Return	Go to start of line 3 lines below cursor
`-`	Go to start of previous line
`3-`	Go to start of line 3 lines above cursor
`(`	Go to start of current sentence
`)`	Go to start of next sentence
Ctrl-d	Scroll down (forward) half a screen
Ctrl-f	Scroll forward by a full screen
Ctrl-u	Scroll up (back) half a screen
Ctrl-b	Scroll back by a full screen
`/`*`expression`*	Find next occurrence of *`expression`*, jump to its first character
`?`*`expression`*	Find last occurrence of *`expression`*, jump to its first character
`n`	Find the next occurrence of *`expression`* (the last search)
`N`	Find the last previous occurrence of *`expression`* (the last search)

Working with Text

The following commands are representative of the many commands available for working with text from the command mode.

`x`	Delete 1 character
`dw`	Delete word
`dd`	Delete 1 line and put it into the buffer
`3dd`	Delete 3 lines and put them into the buffer
`yy`	Yank a line (put current line into buffer)
`3yy`	Yank 3 lines
`J`	Join the next line to the current line
`p`	Insert line(s) from buffer below current line
`P`	Insert line(s) from buffer above current line
`r`	Replace one character by one other character
`u`	Undo the last command
`.`	Repeat the last command
`~`	Change character from upper case to lower case and vice versa
Ctrl-l	Redraw the screen (e.g., remove error messages covering the text)
`ZZ`	Write if necessary and quit `vi`

Changing to the Insert Mode

Any of the following commands enable you to add or replace text. When you finish adding text, press the Esc key to return to the command mode.

`a`	Append text after the current cursor position
`A`	Append text to the end of current line
`i`	Insert text before current cursor position
`I`	Insert text at beginning of current line
`cw`	Change word from current cursor position to end
`c3w`	Change 3 words from current cursor position to end

`c)`	Change sentence from current cursor position to end
`C`	Change line from current cursor position to end
`o`	Open line below current line
`O`	Open line above current line
`R`	Replace (overwrite text character by character)
`s`	Substitute one character

Changing to the Last Line Mode

The last line mode is used to quit `vi`, read and write files, call another file for editing, open a UNIX shell, and similar actions. To change from the command mode to the last line mode, type a colon “:” as the first character of a line.

Insert Mode

A series of commands (listed above) switches you to the *insert mode*, where all the text typed on the keyboard (except for the Esc key) shows up in the text. The only way to exit the insert mode is to press the Esc key, which leads back to the command mode.

Unfortunately, there is no indication on the screen whether `vi` is in the command mode or in the insert mode. Inexperienced operators often press the Esc key to make sure they are in the command mode, but as you become familiar with `vi`, this will be seldom necessary. The Esc key is also the only possibility to avoid the execution of commands that have been partially typed, such as when the number has been typed, but not the last character.

Special (normally not displayable) characters can be inserted into the text if they are preceded by Ctrl-v. For example, pressing Ctrl-v Ctrl-q is displayed in the text as “^Q”.

Last Line Mode

The *last line mode* is also called the `ex` command mode because in this mode you enter commands from the `ex` program:

`:r` *`file`*	Read a file named *`file`*
`:w`	Write back file
`:w` *`file`*	Write under a new file with name given by *`file`*
`:q`	Quit `vi` (quits only if file is unchanged since opened or saved)
`:wq`	Write back and quit `vi`
`:q!`	Quit editor without saving changes
`:n`	Edit next file
`:sh`	Open a shell
`:%s/`*`old`*`/`*`new`*`/g`	Replace all occurrences of string *`old`* by string *`new`*
`:se nu`	Switch on line numbering
`:se nonu`	Switch off line numbering
`:set all`	Show all options
`:e!`	Reedit file, discard changes
`:!`*`commands`*	Execute UNIX commands (*`commands`*) in a temporary shell, return to `vi` afterwards

The `ex` program was an early UNIX line-oriented text editor. For compatibility, `ex` commands were integrated into `vi` as the last line mode, making `ex` a subset of `vi`. With two exceptions, the last line mode is initiated by typing a colon as the first character of a line; thereafter any `ex` command can be entered. The execution of these commands, which requires pressing Return, automatically leads back into the command mode.

Other vi commands

The sections above list only a selection of the most important `vi` commands. For a complete description, refer to the Sun manuals and UNIX textbooks. Many powerful features are available, but they normally are only required for experienced and frequent users. Such features include:

- Options, such as special setups for writing programs.
- Customization file (`.exrc`).
- Macros.
- Abbreviations to avoid retyping long and frequent expressions.
- Additional buffers to increase flexibility in moving text portions around.

For example, the following inline shell calls provide advanced formatting and sorting of text:

`!}fmt -63`	Reformats current text between the cursor and the end of the paragraph to 63 characters per line
`!]]fmt -70`	Reformats current text between the cursor and the end of the file to 70 characters per line
`!}fmt -c -75`	Reformats current text starting at the cursor position (first line) and indents all following lines to the first character of the second line
`!}sort -u`	Alphabetically sorts all lines up to the end of the current paragraph in the text starting at the cursor position, removing duplicate lines.

Problems with Running vi

The `vi` editor formats the screen by moving the cursor around. This can only work if

- The terminal characteristics support such cursor movements
- `vi` knows the exact terminal definition

If there are problems with these points, `vi` either incorrectly formats the screen, making proper editing almost impossible (type ZZ to quit the editor) or switches to “open mode” (it tells you about this on the last line), which essentially is the single line `ex` editor (in this case, type `:q` Return to exit the editor). Other problems are more subtle and relate to shell calls within `vi`, see below.

Using vi in Single-User Mode

Trying to use `vi` in single-user mode can be unsuccessful for several reasons:

- The terminal type is undefined, which makes it impossible for `vi` to format the text screen (or window) properly and control the cursor movements.
- The `PATH` variable might not include `/bin` or `/usr/bin`, such that the `vi` command cannot be found.

There is an easy way out, which gives access to `vi` even in single-user mode (note that this is in a Bourne shell environment—you cannot use `setenv`):

`TERM=sun`	Defines the terminal type.
`export TERM`	Makes `TERM` information available to child processes (e.g. when calling a shell from within `vi`)
`vi`	Calls `vi`

If the `PATH` does not include `/bin` or `/usr/bin`, call `vi` with the full path by entering `/usr/bin/vi`.

Problems with Running vi Remotely

After a remote login through `rlogin` or `telnet`, `vi` often does not work properly because the `term` variable (`TERM` in the Bourne shell) is not defined properly. In particular, the X environment, the `term` variable after a remote login is usually set to `xterm`, which is okay if you are working in an `xterm` window but not in a `shelltool`, `cmdtool`, or `dtterm` (the standard CDE shell window). To correct this, use the following commands:

`set term=dtterm`	For `dtterm` (CDE shell windows)
`set term=sun-cmd`	For `cmdtool` (scrolling OpenWindows shell windows)
`set term=sun`	For `shelltool` (nonscrolling OpenWindows shell windows)

After this, `vi` should function as expected. If you are (remotely) logged in as `root` and/or working in a Bourne shell, use `TERM=dtterm`, `TERM=sun-cmd`, or `TERM=sun` instead.

Problems with Running vi Remotely in CDE

A peculiar problem arises when calling `vi` from a standard CDE window (`dtterm`) while being logged in to a SunOS (and probably any other non-Solaris or non-CDE) system—the proper setting for the `term` environment variable is `dtterm`. Unfortunately, SunOS (`/etc/termcap`) does not contain a definition for the terminal type `dtterm`.

So far, we have not found a good solution for this problem. In Solaris 2.x, `/etc/termcap` only contains the older definitions (for compatibility reasons) but no definition for `dtterm` (we don’t even know whether it would be possible to construct a BSD `termcap` entry for the CDE `dtterm`). On the other hand, we have found two workarounds:

- A simple but only partially functional solution is to temporarily switch to the DEC VT-100 terminal definition by entering `set term=vt100`. For basic `vi` usage, this seems to work reasonably well.
- The better solution is to call an OpenWindows command tool (`cmdtool`) that is available from the program manager (from the CDE toolbar) under “OpenWindows.” You can call the “OW Command Tool” directly or you can drag the icon into the icon drop box of one of the CDE toolbar menus (typically under “Personal Applications”). The OpenWindows command tool uses a terminal definition (`sun-cmd`) that is known both under SunOS 4.1.x and under Solaris 2.x.

Problems with vi Shell Calls in CDE

Occasionally, we have also found problems with `vi` shell calls (through the exclamation mark, see “Other vi commands,” page 145). If you run into this problem, the solution again is to use the OpenWindows command tool (`cmdtool`) rather than the CDE `dtterm`, see above.

In conclusion, it seems that if you want to continue using `vi` rather than the CDE editor `dtpad`, you are probably better off using the OpenWindows command tool (`cmdtool`), rather than `dtterm`.

9.3 Other Editors

The other editors in the standard Sun software are line-oriented in which you work on only one line at a time (apart from printing portions of the text). These editors were basically written for early computer terminals such as the teletype. Under these conditions, editing consisted of entering, modifying. and displaying (printing) single lines.

Obviously, such editors are much more difficult to use than full-screen editors. UNIX provides two of them: `ed` and `ex`.

ed and ex Editors

The `ed` program was the first UNIX text editor. It has a very terse user interface (e.g. a single error message: the question mark) and is not very comfortable.

The `ex` editor is actually a subset of `vi` and, therefore, provides a better user interface (at least one with decent error messages) than `ed`. At the same time, `ex` is an extended and enhanced version of `ed` and was written at the University of California at Berkeley by William Joy.

Line-oriented editors are only used in special circumstances: `ex` can be useful if for some reason only a line terminal is available (which is seldom the case, because most standard RS-232 terminals are supported by `vi`) or if for some reason the terminal setup in UNIX has been corrupted. `ed` is used if for some other reason only a limited UNIX command set is available (e.g., `vi` command file corrupted or erased). Because such cases are rare, these editors are not explained in detail here. A good introduction to both of them is found in the Sun manuals.

emacs Editor

Other editors are available on the market for the Sun. Best known among them is `emacs`, a sophisticated and powerful, screen-oriented editor that many people prefer over `vi`. `emacs` seems to combine the good parts of both `vi` and `textedit`. It is window-oriented, mouse-friendly, modeless, and yet still fast and easy to use. On the other hand, it makes heavy use of the Fn and Rn function keys, and many people don’t like that. Unfortunately, it is not part of the standard Sun software, but Sun user groups can make it available for little or no money because it is public domain software.

Chapter 10. Using Tapes and Floppy Disks

Sections in this chapter:

- 10.1 “tar Command,” this page
- 10.2 “ufsdump, ufsrestore, and dd Commands,” page 155
- 10.3 “Using Floppy Disks,” page 156

OpenWindows includes the Tape Tool and File Manager programs for handling tapes and floppy disks. The OpenWindows desktop utilities can also be called from CDE via the Application Manager. This chapter provides some information about the UNIX commands used by these tools, which is necessary for a full exploitation of those media or for a direct use in a shell window:

10.1 tar Command

The Tape Tool is an interface for selected options of the UNIX `tar` command (derived from “tape archive”). This tool can be used to transfer files by `tar` to or from devices such as a tape drive, floppy disk drive or file. The button labeled props... opens a popup menu for configuring the `tar` command. To fill in the necessary options or to run the `tar` command from a shell, you need to know the structure and features of the `tar` command. We start with tape operations, the most frequent application of the `tar` command.

The `tar` command can be used to read and write files to any kind of tape. It is the standard command for storing directories with all subdirectories on tape. Files that have been written with `tar` must be read back with the same command. `tar` is quite straightforward to use, but it has some drawbacks:

- It does not keep track of the amount of tape used. An attempt to write very large directories to tape starts correctly, but when the end of the tape is reached, `tar` issues an error message and does not ask for further tapes.
- It makes no mapping, therefore reading single subfiles can be time-consuming.

Entire file systems cannot be backed up with `tar`. The `ufsdump`, `ufsrestore` and `dd` commands do file system backup (`ufsdump` uses mapping and can write on multiple tapes where necessary). Chapter 12, “File Security, Repair, and Archiving,” covers this further.

Command Syntax

A typical call to `tar` is

```
tar xvbf 40 /dev/rmt/0mb
```

The `tar` command has a number of single character options, such as `xvbf` used here: `x` for extracting, `v` for verbose showing, `b` for blocking factor, and `f` for source/target file. Unlike most other UNIX commands, the options to `tar` are not preceded by a minus sign.

Some options require an additional argument (e.g., `40` for the option `b`, `/dev/rmt/0mb` for the option `f`). The sequence of arguments *must* correspond to the sequence of these options. `tar xvfb /dev/rmt/0mb 40` is also correct and has the same effect.

`/dev/rmt/0mb` is the location (source or target) of the `tar` file. In this example, the location is a magnetic tape (`rmt` identifies a removable magnetic tape) with logical number `0` and a medium write density `m`.

The final `b` means the device follows the BSD (Berkley Software Distribution); otherwise, the SVR4 (System V, Release 4 of AT&T) convention is used.

Note: If you want to write tapes that are compatible with SunOS 4.x and BDS UNIX, you *must* use the “`b`” device option (`/dev/rmt/0b`, `/dev/rmt/0bn`, `/dev/rmt/0lb`, `/dev/rmt/0lbn`, etc.).

tar Files

The `tar` program is a converter between the UNIX file system (a hierarchical structure) and a `tar` file, and vice versa. A `tar` file is a linear structure and has no directory. You must scan the file itself to find out what its contents are.

In most cases, the `tar` file is located on a magnetic tape. The `f` option indicates the location of the `tar` file. The default value (leaving out the `f` option) is `/dev/rmt/0`, which is taken from `/etc/defaults/tar`. The default can be altered there or with an environment variable `TAPE`. If you want `/dev/rmt/0mb` to be the default device, add

```
setenv TAPE /dev/rmt/0mb
```

to your `.login` file. If the default matches the device you need, you don’t have to specify the `f` option (and the device name itself).

The standard target file for the `tar` command is `/dev/rmt/0`. Having more than one tape drive installed, the only requirement under Solaris 2.x is a different SCSI number. During the boot process (`boot -r`), the tape devices are configured automatically. You will find `/dev/rmt/0`..., `/dev/rmt/1`..., and so on. With `ls -l /dev/rmt`, you can see which of the targets are connected to which device number (...`st@4`...: tape at target 4).

`tar` automatically rewinds the tape at the end of the operation, unless its name is preceded with an `n` (e.g., `/dev/rmt/0n`).

The `tar` file can also be a disk file. Just specify a (plain) disk file, and `tar` creates (or reads from) a `tar` file on the disk:

```
tar cf userlib.tar userlib
```

This is useful for compressing data, because the `compress` command can only compress single files, not complete hierarchical file structures.

A common convention is to use the file name extension `.tar` for `tar` files. As a consequence, do not try to use `tar` with a nonexisting tape device (such as `/dev/rmt/3`). When writing to that “device,” this would eventually create a plain `tar` file `3` in `/dev/rmt` that could easily fill the root partition to 111%, (see “Managing Free Space on Disks,” page 163).

`tar` can also produce output to the standard output or read data from standard input if a minus sign is specified as `tar` file. This is useful for transferring directories (see “Using tar to Transfer Directories,” page 153) or to create `tar` files on the disk:

```
tar cf - userlib > userlib.tar
```

QIC Tape Formats

All Sun tape drives can read quarter-inch cartridges (QIC) in 4- and 9-track mode. Tapes are automatically written in 9-track mode, providing maximum capacity. SPARCstations use different tape drives. They can only write in the 24-track, 150-MB QIC-150 format, but they can also read in the other two formats.

If a tape contains information in 4-track or 9-track format already, the SPARCstation detects that and refuses to overwrite because the tape has the wrong format (even if it is the right tape quality). In such a case, the tape has to be completely degaussed, either with a degaussing coil or by holding the tape to the area with the strongest stray field at the bottom of the NMR magnet, and then removing it slowly while wagging it (flipping it over) rapidly. The fringe field of a 200-MHz magnet may not be strong enough to erase tapes completely. In general, old 25 MB and 60 MB tapes can not be overwritten on a SPARCstation (even after erasing), because the tape characteristics don't match the tape drive.

SPARCstations with 150-MB tape drive require 3M DC6150, DC600-XTD (600 feet, *150 MB maximum*) or equivalent quality tapes for writing on tape. Other tape qualities can be only be read. For newer systems with 2.5-GB QIC tape drive, DC 9250 or equivalent quality tapes are required in order to exploit the full capacity, but DC 6150 tapes can also be used to store up to 150 MB. It is possible that tape drives on different Sun machines have slightly different head adjustments, so that tapes sometimes cannot be exchanged between specific machines.

Blocking Factor

As the name *streaming tape* indicates, cartridge drives operate in *streaming mode*, which means that the drive has to speed up first over a number *s* of sectors before it starts reading or writing a data batch of *b* blocks at full speed. To create a continuous disk file, the tape drive rewinds s sectors from the end-of-file mark and then spools forward *s*+*b* sectors (writing after the end-of-file mark), rewinds *s* sectors, and so on. Obviously, if *b* is the same or smaller than *s*, streaming tape operations are very inefficient. The size of the data batch is determined by the *blocking factor*, which indicates the number of 512-byte data blocks in a data batch. The value is entered as an argument using the `b` option.

The default blocking factor for tape commands is 20. The optimum blocking factor depends on the tape drive and the speed of the disk. For SPARCstations, the optimum is 30 to 40 (QIC-150 tape drives speed up much faster). An optimum blocking factor can speed up tape operations by up to 35% (SPARCstations).

Blocking factors of up to 2000 and more can be selected, but there is be no further gain in speed. Very large blocking factors can eventually lead to an error message

```
could not allocate memory
```

In such a case, try `tar` again with a smaller blocking factor or temporarily exit VNMR first. Because the total number of blocks written to the tape is a multiple of the blocking factor, very large blocking factors are inefficient for writing small amounts of data.

Writing Files

Writing a `tar` file is done using the `c` option, for example,

```
tar cvfb /dev/rmt/0 40 vnmrsys
```

(the `c` option requires the source file to be specified). `tar` can be used to write multiple files in one go. Just enter multiple file names or groups of file names, for example,

```
tar cvbf 40 /dev/rmt/0 file1 file2 file3
tar cvbf 40 /dev/rmt/0 *.fid
tar cvbf 40 /dev/rmt/0 .
```

As mentioned above, `tar` does not check on or keep track of the tape usage. It is up to the user to make sure there is enough space on the tape. Use `du` to check the data size before storing large amounts of data on tape.

`tar` does not follow symbolic links, but it rather writes the symbolic link as such into the `tar` file. This is different from the way the `cp` command works. If absolute paths are specified, `tar` writes absolute names into the `tar` file (see below).

If the verbose option `v` is specified when writing a tape, `tar` prints out (standard output) the character "`a`" ("archiving"), the file name, and the file size in blocks (or information on the symbolic link).

Tape Catalogs

Tape catalogs are obtained with the `t` option, for example,

```
tar tvf /dev/rmt/0
```

The blocking factor does not matter for reading tape catalogs. If the `v` option is also specified, the catalog contains not only the file names, but also information about the permissions, user-ID and group-ID of the file (both IDs in numeric form), the file size (in bytes), and the date of last modification (the date is given in an uniform format, unlike with the `ls` command). Directories are shown with file size 0. A typical `tar` listing (using the `v` option) is displayed as follows:

```
rwxr-xr-x 10/30   0 Jan 26 10:33 1993 bin/
rwxr-xr-x 10/30   0 Jan 25 09:32 1993 vnmrsys/
rwxrwxr-x 10/30   0 Jan 25 09:35 1993 vnmrsys/exp1
rw-rw-r-- 10/30  17 Jan 25 09:35 1993 vnmrsys/exp1/text
...
```

Reading a tape catalog takes considerable time. Because there is no directory, the entire file must be scanned. Therefore, it is advisable to make a catalog once and then to either print it out or (better yet) store that catalog on disk, see also "Verbose Option," page 153.

Reading Files

Reading from a `tar` file is done with the `x` option, for example,

```
tar xvf /dev/rmt/0
```

If the blocking factor specified for reading a tape is different from the factor that was used for writing, this can lead to error messages. To avoid errors, it is recommended to use the same blocking factor for reading as for writing or (better) not to specify the blocking factor when reading tapes.

If no file name is specified when reading from a `tar` file, all files are read back. If selective files should be extracted from the `tar` file, the complete file name must be specified, exactly the way it shows up in the output with the `t` or the `v` options. `tar` does not know wildcard characters for reading, and file names cannot be abbreviated. If a directory is specified, it is extracted with all its subfiles. If only a subdirectory or a subfile is extracted, `tar` automatically generates the necessary directory levels above it. If the file to be extracted already exists on the disk, the disk file is overwritten (provided the user has write permission on it).

If the files were stored with absolute paths, they are also read back the same way. Therefore, storing files with relative paths is highly recommended. Upon extracting, there is no way to redirect files that were stored with absolute path.

Verbose Option

The `tar` program is often used in the verbose mode (`v` option), which can generate considerable output. Running `tar` in a separate window is recommended. If the output is to be inspected during or after the execution of the command, you should either pipe the output to the more command, for example,

```
tar tvbf 40 /dev/rmt/0 | more
```

or use a Command Tool (scrollable window) for scrolling back later on.

Alternatively, the output can be dumped into a text file, for example,

```
tar tvf /dev/rmt/0 > tapecatalog
```

for easy and fast scanning of tape contents. Using the `grep` command, catalogs from dozens of tapes can be scanned for a specific file name in a fraction of a second.

Making Things Simpler

It is recommended that you use the environment variable `TAPE`. This allows you to leave out the `f` option in most cases. To simplify operation using `tar`, you can also define the following aliases in `.cshrc`:

```
alias twrite tar cvbf 40 /dev/rmt/0
alias tcat tar tvf /dev/rmt/0
alias tread tar xvf /dev/rmt/0
```

This makes the short commands `twrite`, `tcat`, and `tread` available for tape operations.

The Tape Tool in Open Windows allows to do `tar` read/write operations in an interactive way using a simple graphical interface that opens options such as `c`, `x`, `t`, `b`, and (mandatory) `f` for the source/target. The `f` option extents Tape Tool to a tool for transferring files also to floppy disks or other files (file systems).

Using tar to Transfer Directories

You can also use `tar` to copy directories (with subfiles) between file systems without using a tape (see the online manual or the Sun command reference manual). For example:

```
su
cd /export/home
mkdir vnmr.bk
cd /vnmr
tar cf - * | (cd /export/home/vnmr.bk; tar xvfBp -)
```

All files and directories (including subdirectories) in `/vnmr` are transferred to the standard output (`-`), then piped (`|`) to the standard input where a second `tar` process, which is started in `/export/home/vnmr.bk` beforehand, reads all incoming files and saves them there in the original form. The `B` option adjusts the blocking factor to the standard input, and the `p` option tries to keep ownerships, permissions, and modification dates the same as in the source files (the ownership part of this option only works if this is done as root). This sounds much the same as `cp`, but there is a subtle difference between the two commands—`cp` does not copy symbolic links but rather the file itself which the link is pointing at.

Using Multifile Tapes

`tar` automatically rewinds the tape at the end. Therefore, the `tar` command cannot be applied twice on the same tape without erasing the first data (only one write operation per tape is possible) unless special measures are taken.

A data segment generated with a single tape command can be skipped with a magnetic tape control command

```
mt -f /dev/rmt/0n fsf 1
```

where the `n` stands for no rewind and the `1` is the number of data segments to be skipped. `tar` can also be called with a "no rewind" option by using the device name `/dev/rmt/0n`, for example,

```
tar cvbf 40 /dev/rmt/0n *
```

But, in general, this practice is not recommended because the access to the second (or a further) data segment is rather awkward. A simple tape catalog (`tar tvf /dev/rmt/0`) only looks at the first data segment and does not indicate the existence of further data on the tape. For most read or catalog operations, the `mt` command has to be called for skipping the required number of data blocks.

Further instructions on how to handle multisegment tapes, and a shell script for obtaining complete multisegment tape catalogs were published by S. Patt and E. Williams of Varian, Palo Alto, in *Magnetic Moments*, Vol. IV, No. 1, pp. 14 -15. An updated version of that script is available with `bin/archive` from the VNMR user library.

Using tar Remotely

Frequently, a tape drive is only available on a remote system in the local network (see Figure 21).

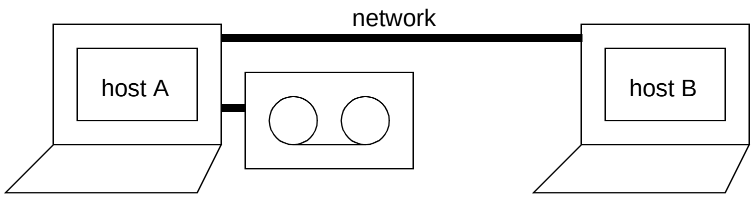


Figure 21. Local Network with Tape Drive

So how can you transfer data to or from this tape drive via the network without temporarily storing data on the disk of the tape server? The solution is to transfer the unextracted `tar` file via the network. On the system with the tape drive, you call `dd` to read or write the `tar` file to or from the tape drive and then you use a UNIX pipe to link the `dd` input (or output) with a `tar` command that is called on the system with the disk.

Reading Data from a Remote Tape Drive

Reading data from a remote tape drive can be initiated from both hosts involved. If you are working on host A, you can use the following command line:

```
dd if=/dev/rmt/0 ibs=20b | rsh B "(cd dir; tar xvf - files)"
```

to read the specified files, or

```
dd if=/dev/rmt/0 ibs=20b | rsh B "(cd dir; tar xvf -)"
```

if you want to read all files. The `dd` option `ibs=20b` (or whatever block size is selected) only needs to be specified if the tape was written with a nonstandard blocking factor, or when the blocking factor was specified when writing the tape). If you are logged into host B, you can use the commands

```
cd directory_name
rsh A dd if=/dev/rmt/0 ibs=20b | tar xvBfb - 20 files
```

(you are now executing the `dd` command remotely, rather than `tar`).

Writing Data to a Remote Tape Drive

Writing data to a remote tape drive can be performed from both hosts involved. If you are working on host A, you can use the following command line:

```
rsh B "(cd dir; tar cvf - files)" | dd of=/dev/rmt/0 obs=20b
```

It is not really necessary to specify the blocking factor (option `obs=20b` with the `dd` command), but if you do so, you need to specify the same blocking factor when reading that tape. If you are logged into host B, you can use the commands

```
cd directory_name
tar cvf - files | rsh A dd of=/dev/rmt/0 obs=20b
```

More information on the UNIX commands `rsh` and `dd` can be found in the UNIX manuals (use the `man` command).

10.2 ufsdump, ufsrestore, and dd Commands

The `ufsdump`, `ufsrestore`, and `dd` commands are used almost exclusively for backing up and restoring disks, which is part of the system administration. Chapter 12, "File Security, Repair, and Archiving," discusses this in detail.

`dd` can also handle byte-swapping where necessary (on certain computer systems the byte-order is reversed as compared to the Sun).

Reading Gemini and VXR-4000 Tapes

`dd` is also a general-purpose command for reading tapes of any format, including the format used with other computer systems such as Gemini and VXR-4000 systems.

VXR-4000 tapes created with the command `STAPE(WRITE,DSKn,...)` consist of two files: a 1-sector (512 bytes) header file and the data file. Reading such tapes is as simple as skip one file and then read the next file. The VNMR software contains the `tape` utility that reads such tapes. Unfortunately, `tape` is very slow in reading 1/4-inch tape cassettes, because it reads the data sector by sector (like using `tar` with a blocking factor of 1). Therefore, the following procedure might be useful as an alternative:

`mt -f /dev/rmt/0lbn rewind`	Rewind tape
`mt -f /dev/rmt/0lbn fsf 1`	Skip tape header
`dd if=/dev/rmt/0lbn of=tape.288 ibs=512000`	Read the second tape file
`mt -f /dev/rmt/0lbn rewind`	Rewind tape
`decomp tape.288`	Decompose tape file

In this example, a blocking factor of 512000 bytes (1000 blocks) was selected. This size speeds up the reading operation dramatically.

The name of the output file can be selected freely except for the suffix, which must be equal to the maximum number of directory entries (I2 for the file FCB on the Gemini or VXR-4000). Otherwise, the `decomp` command, which splits up a VXR-4000 file into its subfiles, cannot work properly.

Disk units on the VXR-4000 always have 288 entries. Other directories (e.g. an automation file or user-created files) likely have a different number of entries. Of course, `tape` extracts that information from the tape header. If you are not sure about the number of directory entries, you might want to try out different values of the suffix:

`rm -r tape`	Remove previously decomposed file
`mv tape.288 tape.144`	Select a different suffix
`decomp tape.144`	Decompose again

You only need to try multiples of 36 as suffix. Because `decomp` does not alter the source file, you can try many times. The VNMR command `tape` has one advantage—it allows *selective* file extraction from a VXR-4000 tape, whereas with the above procedure, the entire tape (which can be as large as 16 MB) must be read onto the disk.

10.3 Using Floppy Disks

All desktop SPARCstation and Ultra models can be equipped with a 3.5-inch floppy disk drive that can read and write floppy disks in the DOS format (PCFS), UNIX file system format (UFS), or `tar` format. Again, for most operations, the Open Windows File Manager can handle the PCFS and UFS formats, and the Tape Tool the `tar` format.

This section describes these operations in detail, starting with UNIX commands and referring to the tools mentioned above.

Formatting Floppy Disks

Before they can be used under DOS or UNIX, floppies must be formatted into the UFS or DOS format. Note that the SPARCstation recognizes high-density floppies by the additional square hole at the rear edge of the floppy disk and refuses to format such floppies in low-density mode (of course, the opposite is also true). The raw size of formatted floppies is 1.44 MB for high density (with the extra hole in the corner) or 720 KB for double density.

To format a floppy, insert it into the drive (remove the write protection first by making sure the corner hole is covered). Then enter `fdformat` for a high-density floppy (1.44 MB) or `fdformat -l` for a low-density floppy (720 MB). The DOS format is provided if a `-d` option is used. To eject the floppy after formatting, add the `-e` option to the `fdformat` command or enter the command `eject` after the formatting is finished. The `-v` option causes `fdformat` to verify the floppy after formatting.

In case the floppy disk has a raw format, the above procedure is straightforward. If it has a DOS or UNIX file system format, the volume manager takes over the control and blocks any operation on a floppy from a shell. A superuser can kill the volume manager, format the disk, and can start the manager again (see "Floppies with the tar File System," next). A more direct way is to add the `-U` option for `fdformat`, which unmounts any file system. This can be done by any user. To mount it again, eject the disk by the `eject` command and insert it again.

The File Manager also provides a format option. It turns out, however, that this tool cannot alter the format of a floppy disk. It denies reformatting a DOS floppy into a UNIX formatted floppy disk, and vice versa. Formatting is only possible using an unformatted or `tar` formatted disk.

A trick overcomes this handicap. In most cases, a new floppy disk is preformatted with the DOS file system. If you want to use a UNIX file system instead, you can “unformat” the disk by destroying the format in the stray field of a supercon magnet. In such a case, hold the disks to the area with the strongest stray field at the bottom of the NMR magnet, and then remove it slowly while wagging it (flipping it over) rapidly. After this, the File Manager formats the disk.

Floppies with the tar File System

The most direct way to use a floppy disk is with the `tar` command. The disk must be preformatted (UNIX or DOS). In principle, there are the same options and arguments as for tape operations. Typically a command used has the form:

```
tar cvf /dev/diskette file1,file2,...
```

Instead of *`diskette`*, other names such as `floppy` or `rfd0a` ... `rfd0e` could be defined in `/dev` pointing to the same or similar floppy disk drives. The `r` stands for raw (no file system is expected), `fd` for floppy disk, and `a` through `e` determine which cylinders are used (see the UNIX manual). Using `diskette` is highly recommended, because it guarantees the best exploitation of the floppy disk.

The Tape Tool can also be used to store files on a floppy disk with `tar`. The operation is similar to that with a tape. However, the device name has to be changed to, for example, `/dev/diskette`.

To use Tape Tool or the `tar` command from a shell, you to get access to the floppy disk drive. It is usually occupied by the volume manager. Trying to execute `tar` results in an error message:

```
tar: /dev/diskette: Device busy.
```

To get access, you need to kill the `vold` process:

```
su root -c '/etc/init.d/volgmt stop'
```

With `start` instead of `stop`, the management is started back up again.

Alternatively, you can also just kill the process named “`vold`”. First, get the process-id:

`ps -ef | grep vold`	Gives 3-digit id-number of the process

and kill it as a superuser by *one* of the following:

`su; kill` *`id-number`*`;` [Ctrl-d]	Kill process in one step
`su root -c 'kill` *`id-number`*`'`	As `root`, kill the process

To have the volume management back (e.g., to run a floppy disk drive with File Manager) you must start it again as a superuser:

```
su root -c '/usr/sbin/vold &'
```

Floppies with DOS Format

For transferring data to a PC or a Macintosh, the floppy drive in desktop SPARCstations can also handle floppies with a DOS file system. Such floppies can be formatted on a PC or on the Sun. Use `fdformat -U -d` to format the floppy and also create a DOS file system on it.

The easiest way to handle DOS formatted floppy disks is via the File Manager of Open Windows. It automatically recognizes the format and transfers files between UNIX and DOS by mouse clicks.

For a manual mounting/unmounting you have to do the following:

To mount it, we create a directory `/pcfs` and add a line to the file `/etc/vfstab`:

```
/dev/diskette - /pcfs pcfs - no -
```

The floppy can now be mounted with the command

```
 mount /pcfs
```

Any user can change directory into the new disk and create, copy, modify and delete files, directories and subfiles (a hierarchical file system).

To eject the floppy, you must first change the working directory (the floppy cannot be unmounted while somebody has their working directory in it), then unmount and eject the floppy with the commands:

```
umount /pcfs
eject
```

Unfortunately `mount` and `umount` can only be done by `root`, which is a security drawback with this method of using the floppy drive.

The DOS file system will have 1423 KB of free space (the boot block uses up some space). The floppy is optimized for space, *not speed*. To write a single 1-MB file onto the floppy can easily take over 10 minutes.

The DOS file system is also not optimized towards storing many tiny files. Trying to store all VNMR macros on a floppy stops after 225 files, because the directory space is exhausted (although only 250 KB are stored on the floppy). The disk directory is not allowed to contain more than 225 subfiles. All macros *can* be stored in a directory *within* the floppy, but it takes about 90 minutes to write 500 macros (compared to about a minute with `tar`), and even then it doesn't work properly, due to the file name restrictions, see below.

In some aspects, the DOS file system almost behaves like a UNIX file system, but nevertheless it has to follow DOS conventions: file names are truncated to 8 characters and any extension to 3 characters. Lowercase characters are mapped into uppercase (`ls` shows all names in lowercase). This is another severe limitation for any attempt to use a DOS floppy like a generic UNIX file system.

In addition to that, there are no permissions and no owner-ID or group-ID in a DOS directory, of course. `ls` shows all files with `rwxrwxrwx` permission, and all files are owned by the user `root` and the group `daemon`.

For transferring text files to a DOS format, you should make sure that the text itself fulfills DOS conventions, such as regarding `cr` and `lf` characters. The Sun software contains utilities to convert files in both directions:

```
unix2dos -ascii unix_file dos_file
dos2unix -ascii dos_file unix_file
```

For Sun/UNIX text file, it is advisable to use the `-ascii` option, and eventually also the -7 option, but this may need to be tried out, depending on the target application program on the PC or Macintosh.

Floppies with UNIX File System

Another possibility with floppy disks is to create a UNIX file system on them after formatting. This makes the floppy identical to the other disks in every aspect (except for size and speed, of course). You gain a fully hierarchical file system, within which we can freely create, copy, delete, and modify files. There are also no DOS limitations. A large number of files can be stored without restrictions in the file names and with the full UNIX file protection scheme (permissions, ownership), the same as on a hard disk.

Not only do you gain additional flexibility, but also a great deal of performance over the DOS file system. When storing a plain 1-MB file *onto* the floppy, UNIX returns the control to you after just two to three seconds. The data transfer finishes in background after 50 seconds (UNIX data transfers *to* the disk are buffered; `tar` does it in 36 seconds, the DOS file system takes 11 minutes). Storing the VNMR macros or the entire `maclib` directory is no problem, although 500 macros take about 9 minutes to store (45 seconds with `tar`, about 90 minutes with the DOS file system).

The easiest way is to use the File Manager. The UNIX formatted disk is mounted automatically. The floppy disk can be used as any other disk. Any user is able to do this.

What is necessary to know about and what happens in the mount and unmount process?

This can be understood best going the manual way in a shell window while the Volume Manager `vold` is not running. (See above for how to stop `vold`).

The UNIX file system is generated with the command

```
 newfs /dev/diskette
```

(note that we have to specify a partition, not a disk). After this the disk can be mounted on an empty directory using the commands

```
mkdir /floppy
mount /dev/diskette /floppy
```

and now the floppy can be used. We can change a directory into it, create directories, store and edit files on it, like with any other file system. The only problem is that mounting can only be done by `root`, and before it can be ejected, it has to be unmounted again by `root`. Note that here the floppy also uses the standard UNIX permissions and ownership to protect the data, different from a DOS file system.

The following is a typical sequence of commands without the File Manager for using a new floppy containing a UNIX file system:

```
fdformat
newfs /dev/diskette
mount /dev/diskette /floppy
...
(use floppy)
...
umount /floppy
eject
```

As mentioned above, note that you have to be `root` to mount and unmount the disk, and also that unmounting is only possible if nobody is using the floppy as working directory. The `newfs` command gives the following output:

```
/dev/rdiskette: 2880 sectors in 80 cylinders of 2 tracks, 18
sectors
        1.4MB in 5 cyl groups (16 c/g, 0.28MB/g, 128 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 640, 1184, 1792, 2336,
```

If we use `df` to see how much space there is on the floppy, we get the following output:

```
/dev/diskette     1263      9     1134     1%      /floppy
```

Obviously, a part of the capacity of the disk has been lost in the creation of features necessary for the UNIX file system. The data are organized in 8-KB blocks, the disk is organized in 5 cylinder groups to make the file system fast, every cylinder group has a spare superblock for recovering from disk problems, and the disk reports 100% full when the filling degree is only 90%. We don't need all these features, because the floppy is slow anyway, and we don't need all the backup provided for hard disks.

To store more data on a floppy, use 4 KB instead of 8 KB block size, reduce the fragment size from 1 KB to 512 bytes, use 1 cylinder group of 80 sectors (covering the entire disk) only, and allow for 100% disk usage instead of 90%. The command:

```
varian# newfs -b 4096 -f 512 -m 0 -c 80 /dev/diskette
setting optimization for space with minfree less than 10%
newfs: /dev/rdiskette last mounted as /floppy
newfs: construct a new file system /dev/rdiskette: (y/n)? y
/dev/rdiskette: 2880 sectors in 80 cylinders of 2 tracks, 18
sectors
   1.4MB in 1 cyl groups (80 c/g, 1.41MB/g, 704 i/g)
super-block backups (for fsck -F ufs -o b=#) at
  32,
```

You could further increase the space by decreasing the number of I-nodes, but this may limit the number of files that can be put onto the floppy.

`newfs` also creates a directory `lost+found` in the new file system, for the file system check command `fsck` to store recovered files. If you don't use `fsck` on the floppy, you can also delete that directory. If `df` is now used to see how much space is on the floppy, you get the following output:

```
/dev/diskette       1323        4     1319    1%   /floppy
```

This has recovered more than 10% of disk space over the standard setup (we have only lost 10% of the `tar` floppy disk capacity. Note that although we have deliberately not optimized the file system for speed—it is still much faster than with the DOS file system. The above performance tests were performed with the space-optimized file system.

CAUTION: **Only experts should apply this or a similar procedures to improve the capacity or the performance of hard disk partitions. In general, it is not advisable to change the defaults for hard disks (in special cases, you can use `tunefs` for file system optimization either for speed or for capacity, see the Sun manuals or enter `man tunefs`).**

Of course, mounting can be simplified by creating an appropriate entry in `/etc/vfstab` (see "Automatic Mounting (/etc/vfstab)," page 238). For example:

```
/dev/diskette - /floppy  ufs - no -
```

Mounting can now be done by entering

```
mount /floppy
```

The above line in `/etc/vfstab` can coexist with the line for mounting a floppy with DOS file system, because nothing is mounted automatically.

Transferring Large Files Via Floppy Disks

Occasionally, there is a need to store and transport large files via floppy, such as when a VNMR patch should be loaded on a system that is not networked but is equipped with a floppy drive. Some VNMR patches (such as the patch `6.1AgenSOLino101.tar.Z` for

UNITY*INOVA* systems running VNMR 6.1A in the example below) exceed the available disk space (1.44 MB) on a floppy disk.

The `cpio` command opens up a possibility to store such large files on multiple floppies. In order to use `cpio`, you must first (as `root`) stop volume management using the command

```
/etc/init.d/volmgt stop
```

Now, you can insert a formatted floppy and enter:

```
echo 6.1AgenSOLino101.tar.Z | cpio -oc > /dev/rdiskette
```

After a while, the system ejects the floppy and you get the output

```
End of medium on "output".
To continue, type device/file name when ready.
```

At this point (and whenever this output turns up again), insert a new (formatted) floppy and type (but do not press Return because it will cause the command to abort):

```
/dev/rdiskette
```

At the end of the data transfer, the system reports the total number of blocks, and you can eject the last floppy with the command

```
eject diskette
```

For important files, it is advisable to make a second copy. At the very least, you should immediately verify whether the file can be read back completely, using the procedure given below. Finally, you can restart the volume management as `root`:

```
/etc/init.d/volmgt start
```

The procedure to read the file back from the floppies is similar (you again must disable the volume management for the duration of this task):

```
varian> cpio -ic < /dev/rdiskette
End of medium on "input".
To continue, type device/file name when ready.
/dev/rdiskette
End of medium on "input".
To continue, type device/file name when ready.
/dev/rdiskette
7671 blocks
   varian> eject diskette
   varian> ls -l
total 7696
-rw-r--r--  1 vnmr1  nmr 927375 Dec  5 12:00
6.1AgenSOLino101.tar.Z
```

Reading bar Floppies

In SunOS 4.x, files could be written to floppy disks using the `bar` command. In contrast to `tar`, `bar` allowed adding files to an existing `bar` file on the floppy, and it also permitted storing big files on multiple floppies. `bar` is no longer supported in Solaris 2.x, and you should use `tar` or `cpio` (for large files, see above) instead. `cpio` also permits reading `bar` floppies that were generated under SunOS 4.x. To do this, you must first stop the volume management as `root`:

```
/etc/init.d/volmgt stop
```

Then you can insert and read the `bar` floppy, and then eject it using `eject`:

```
cpio -iH bar < /dev/rdiskette
eject diskette
```

After the `cpio` step, you can restart it again (as `root`) using
`/etc/init.d/volmgt start`

Chapter 11. Periodic Maintenance and Troubleshooting

Sections in this chapter:

- 11.1 “Daily and Periodical Care of Computer Hardware,” this page
- 11.2 “Managing Free Space on Disks,” page 163
- 11.3 “Automatic File System Maintenance,” page 164
- 11.4 “Troubleshooting,” page 165

Although Sun systems require almost no periodic maintenance, there are several things you can do to keep your system running smoothly and reduce the chance of a breakdown. This chapter should help.

11.1 Daily and Periodical Care of Computer Hardware

Little hardware maintenance is required with Sun systems:

- Use `lockscreen` from the `root` menu to avoid screen burn-in effects. Under CDE, an automatic screen blanker is activated by default. Using `lockscreen` is only needed when you want password protection while the screen is locked. On SPARCstations and Ultra workstations, switching off the screen overnight (to save energy) should be no problem and is recommended.
- External tape drives and CD-ROM drives are normally used only from time to time. They should be switched off in-between to save energy and to stop dust collection through the cooling fan.
- When necessary, degauss color screens by switching it off and on again (monitors that are permanently exposed to strong magnetic fringe fields may require degaussing with an external degaussing coil).
- From time to time, clean the screen using a soft cloth and the appropriate cleaning fluid (see the manual for your monitor).
- Periodically check the fans (where applicable) and clean the air filters.
- Clean the head of the streaming magnetic tape drive periodically using a cleaning tape.

11.2 Managing Free Space on Disks

Software maintenance on a multiuser system primarily means file system discipline. Check the file systems periodically with the `df -k` command. For safety reasons, `df -k` reports “100% full” at the 90% level. “Completely full” is actually 111%. Whenever a file system is getting full, put unnecessary data files onto tape and/or remove them.

Use `du -sk *` to check for large files and directories (output is in kilobytes). When file systems are filled up, you most likely detect a drastic slowdown of most UNIX software. In severe cases, individual commands or even UNIX might crash or hang up.

Here are some other suggestions on maintaining free disk space:

- When programs crash (or similar serious things happen), UNIX usually dumps a `core` file (the current program-related contents of the RAM) into the present working directory for later debugging. Because no one is likely to be able to do such debugging, you can safely remove the `core` file, which might be more than 1 MB. Try to educate all users to work in their home directory, whenever possible. If they must temporarily work in some other directory, they should use `cd` to go back into their home directory. Automatic removal of core files can be done using `cron` (see below). A single line

  ```
  limit coredumpsize 0
  ```

  in `~/.cshrc` (also in `/.cshrc` for `root`) stops the system from dumping core files—a much better solution.
- As `root`, you can easily get the `root` partition "111% full" by submitting the wrong arguments to tape commands such as `tar` or `ufsdump`. When the system tries to write onto a nonexisting tape device (e.g., using `tar` as `root` and incorrectly typing `/dev/rmt/9mb` instead of `/dev/rmt/0mb`, or typing a correct device but having it not configured), the data actually ends up in the corresponding file in `/dev/rmt`. There is normally only limited free space on the root partition; therefore, it is easy to fill it up. Only limited operations are possible with a full root partition, but file systems can still be inspected and files can be deleted (if not in multiuser mode, then in the single-user mode). To fix the problem, remove unwanted files. To find those or other erroneous files, enter `ls -l /dev/rmt|more`. All correct files should be symbolic links to `/devices`, so erroneous additional files can be easily recognized.

Of course, software maintenance also means maintaining software security by having files backed up on tape at the appropriate intervals. Use passwords to protect your software and data from hackers and others that could destroy or alter it.

11.3 Automatic File System Maintenance

The useful UNIX feature `cron` can be used to periodically search the file system for obsolete files and reduce chronically growing files to a reasonable size. `cron` looks up a file `/var/spool/cron/crontabs/root` every minute. Periodic actions can be specified to be performed at a certain minute of the hour, a certain time of the day, a certain day of the week, and so on.

To find and remove core files from the `/home` directory on a weekly basis, such as every Saturday at midnight, the following `crontab` file entry may be used:

```
0 0 * * 6 find /export/home -name core -exec rm {} \;
```

The first `0` stands for the minute, the second `0` for the hour, the third entry is the day of the month (`*` indicates any), the fourth entry is the month, the next entry stands for the day of the week (`0` is Sunday, `1` is Monday, etc.).

Every user can create a separate `crontab` file. Such a file is created and edited with the command `crontab -e`. A printout (without editing) can be obtained by `crontab -l`. Never try to edit the `crontab` file directly, always use the `crontab` command because otherwise the new entry is not activated.

A shell uses the uncommon line editor `ed` as default for `crontab -e`. To get the `vi` editor (as superuser), enter the command `EDITOR=vi; export EDITOR` before executing `crontab -e`. For a superuser, this can be made permanent by putting those commands into the file `.profile` under the directory `/`. For any other user, put the command `setenv EDITOR vi` into the `.login` file of the user's home directory.

11.4 Troubleshooting

This section presents some general troubleshooting help including acquisition status codes, Ethernet network problems, system hanging, and other topics. Be sure to contact Varian service for expert maintenance and repair of your system.

Acquisition Status Codes

Whenever `wbs`, `wnt`, `wexp`, or `werr` processing occurs, the acquisition condition that initiated that processing is available from the parameter `acqstatus`. This acquisition condition is represented by two numbers, a "done" code and an "error" code. The done code is set in `acqstatus[1]` and the error code is set in `acqstatus[2]`. Macros can take different actions depending on the acquisition condition.

The done codes and error codes are listed below and in the file `acq_errors` in `/vnmr/manual`. For example, a `werr` command could specify special processing if the maximum number of transients is accumulated. The appropriate test would be:

```
if (acqstatus[2] = 200) then
“do special processing, e.g. dp='y' au”
endif
```

These codes apply to all systems, except codes marked with an asterisk (*) are not used on *MERCURY* and *GEMINI 2000* systems.

Done codes:

11. FID complete
12. Block size complete (error code indicates `bs` number completed)
13. Soft error
14. Warning
15. Hard error
16. Experiment aborted
17. Setup completed (error code indicates type of setup completed)
101. Experiment complete
102. Experiment started

Error codes:

Warnings

101. Low-noise signal
102. High-noise signal
103. ADC overflow occurred
104. Receiver overflow occurred*

Soft errors

200. Maximum transient completed for single precision data
201. Lost lock during experiment (LOCKLOST)

300. *Spinner errors*:
301. Sample fails to spin after 3 attempts to reposition (BUMPFAIL)
302. Spinner did not regulate in the allowed time period (RSPINFAIL)*
303. Spinner went out of regulation during experiment (SPINOUT)*

395. Unknown spinner device specified (SPINUNKNOWN)*
396. Spinner device is not powered up (SPINNOPOWER)*
397. RS-232 cable not connected from console to spinner (SPINRS232)*
398. Spinner does not acknowledge commands (SPINTIMEOUT)*

400. *VT (variable temperature) errors:*
400. VT did not regulate in the given time `vttime` after being set
401. VT went out of regulation during the experiment (VTOUT)
402. VT in manual mode after auto command (see Oxford manual)*
403. VT safety sensor has reached limit (see Oxford manual)*
404. VT cannot turn on cooling gas (see Oxford manual)*
405. VT main sensor on bottom limit (see Oxford manual)*
406. VT main sensor on top limit (see Oxford manual)*
407. VT sc/ss error (see Oxford manual)*
408. VT oc/ss error (see Oxford manual)*
495. Unknown VT device specified (VTUNKNOWN)*
496. VT device not powered up (VTNOPOWER)*
497. RS-232 cable not connected between console and VT (VTRS232)*
498. VT does not acknowledge commands (VTTIMEOUT)

500. *Sample changer errors:*
501. Sample changer has no sample to retrieve
502. Sample changer arm unable to move up during retrieve
503. Sample changer arm unable to move down during retrieve
504. Sample changer arm unable to move sideways during retrieve
505. Invalid sample number during retrieve
506. Invalid temperature during retrieve
507. Gripper abort during retrieve
508. Sample out of range during automatic retrieve
509. Illegal command character during retrieve*
510. Robot arm failed to find home position during retrieve*
511. Sample tray size is not consistent*
512. Sample changer power failure during retrieve*
513. Illegal sample changer command during retrieve*
514. Gripper failed to open during retrieve*
515. Air supply to sample changer failed during retrieve*
525. Tried to insert invalid sample number*
526. Invalid temperature during sample changer insert*
527. Gripper abort during insert*
528. Sample out of range during automatic insert
529. Illegal command character during insert*
530. Robot arm failed to find home position during insert*
531. Sample tray size is not consistent*
532. Sample changer power failure during insert*
533. Illegal sample changer command during insert*
534. Gripper failed to open during insert*
535. Air supply to sample changer failed during insert*

593. Failed to remove sample from magnet*
594. Sample failed to spin after automatic insert
595. Sample failed to insert properly
596. Sample changer not turned on
597. Sample changer not connected to RS-232 interface
598. Sample changer not responding*

600. *Shimming errors:*
601. Shimming user aborted*
602. Lost lock while shimming*
604. Lock saturation while shimming*
608. A shim coil DAC limit hit while shimming*

700. *Autolock errors:*
701. User aborted (ALKABORT)*
702. Autolock failure in finding resonance of sample (ALKRESFAIL)
703. Autolock failure in lock power adjustment (ALKPOWERFAIL)*
704. Autolock failure in lock phase adjustment (ALKPHASFAIL)*
705. Autolock failure, lost in final gain adjustment (ALKGAINFAIL)*

800. *Autogain errors*.
801. Autogain failure, gain driven to 0, reduce `pw` (AGAINFAIL)

Hard errors
901. Incorrect PSG version for acquisition
902. Sum-to-memory error, number of points acquired not equal to `np`
903. FIFO underflow error (a delay too small?)*
904. Requested number of data points (`np`) too large for acquisition*
905. Acquisition bus trap (experiment may be lost)*

1000. *SCSI errors:*
1001. Recoverable SCSI read transfer from console*
1002. Recoverable SCSI write transfer from console**
1003. Unrecoverable SCSI read transfer error*
1004. Unrecoverable SCSI write transfer error*

1100. *Host disk errors:*
1101. Error opening disk file (probably a UNIX permission problem)*
1102. Error on closing disk file*
1103. Error on reading from disk file*
1104. Error on writing to disk file*

Converting References to Subnet 10.0.0 (UNITY*INOVA*)

Prior to VNMR 5.2F, the UNITY*INOVA* console was placed on subnet 120.0.0. In compliance with RFC1597, the UNITY*INOVA* console now uses subnet 10.0.0. The UNIX command `setacq` automatically converts references from 120.0.0 to 10.0.0 in `/etc/hosts` and `/etc/bootptab`.

If it becomes necessary to switch back to an older version of VNMR, be aware that it is *not possible* to reverse the subnet change using versions of `setacq` before VNMR 5.2F. Only

`/etc/bootptab` will be correct after an older version of `setacq` is run. To correctly run an older version of VNMR, use a text editor to change the two entries for `inova` and `inovaauto` in `/etc/hosts` from 10 to 120.

Display Hardware Console Status (UNITY*INOVA)*

Use the UNIX command `ihwinfo` to display the status of digital hardware in the UNITY*INOVA* console. The command

```
ihwinfo startup
```

displays the status at the conclusion of the last console startup (powerup, reboot, etc.). The command

```
ihwinfo abort
```

displays the status the last time an acquisition was aborted or the console rebooted from the host computer (`abortallacqs`). In this context, exiting from either the FID display or lock display of `acqi` counts as an abort. Only the status from the last abort can be displayed.

Global Parameter File Corrupted

Symptoms that the global parameter file (`~/vnmrsys/global`) is corrupted include messages such as the following:

- `variable printer not found`
- `variable wcmax not found`
- `variable lastmenu not found`

The most frequent cause for this problem is probably a full disk when exiting VNMR, but it is also caused by killing VNMR from exiting the window environment directly.

To correct the problem:

1. Exit VNMR.
2. Enter:
   `cd /vnmr/user_templates/global ~/vnmrsys`
3. Restart VNMR.
4. Reselect your printer and plotter.

To avoid global parameter file corruption in the future, follow these precautions:

- Always exit from VNMR before exiting from the window environment.
- When the disk is full, do not exit VNMR but free up some disk space first.
- Always exit VNMR by using the Exit VNMR button or by using the exit command in VNMR.
- Do not start VNMR with a full `/home` (or `/export/home`) partition or with a full user disk quota.

Insufficient Swap Space

If the swap space on the host computer is set too low, processes such as PSG may not find sufficient space in which to operate. For proper VNMR operation, the disk should have been partitioned with swap space equal to twice the computer memory. On systems using Solaris 2.x, you can enter the command `swap -l` to check the amount of free swap space.

If the proper amount was not partitioned, you can increase the available swap space, as follows:

1. Become `root`.
2. Use the `mkfile` command to make a new file for the additional swap space. The size, name, and location of this file are up to you (e.g., for a 16-MB file named `swap` in `/export/home`, enter `mkfile 16m /export/home/swap`).
3. Enter `swap -a /export/home/swap` to add the file to the available swap space.
4. To make the new swap space permanent so that it reappears when the computer is rebooted, edit the file `/etc/vfstab` and add the following line:
   ```
   /export/home/swap    -    -    swap    -    no    -
   ```
   where the first entry is the name of the file created in step 2 above.

To remove a swap space, use the `swap -d` command to release the swap space, followed by the `rm` command to remove the swap file.

tmp Partition Fills Up

If the `/tmp` partition fills up, it can cause problems for VNMR users. One solution is to move the `tmp` partition. Here is one method for doing so, which assumes that you have lots of free space in a different partition (`/data` in this example).

1. Make sure there is no user activity on the system.
2. Enter:
   ```
   su
   cd /
   tar cf - tmp | (cd /data; tar xfBp -)
   rm -r tmp; ln -s /data/tmp tmp
   ```
3. Reboot UNIX.

Password Invalid

A password may be invalid because it has expired. Solaris 2.x has a password aging mechanism that disables passwords after a predefined time. This security feature can be activated to force users to periodically change their passwords. `root` can turn on password aging for an existing user (like `vnmr1`) by entering:

```
passwd -n 7 -x 1000 -w 30 vnmr1
```

This sets the time between password changes by `vnmr1` to 7 days, the time the password is valid to 1000 days, and the time when warning messages start on login to 30 days prior to the expiration date. After expiration, the `vnmr1` password becomes invalid and can only be reset by `root` using the command

```
passwd vnmr1
```

Password aging can also be entered by `root` through the `admintool` program.

While password aging may be desirable for standard users, especially in larger networks, it is not appropriate for users such as `acqproc` that usually have no password (even users with no password can become invalid if password aging is turned on). Should password aging be inadvertently turned on, commands such as `su acqproc` may become invalid.

To tell the system that a user (such as `acqproc`) will never again need a password, `root` should enter:

```
passwd -d acqproc
```

To disable password aging for any user after VNMR is installed, `root` should enter: `passwd -x -1 acqproc.`

Password aging for `root` should be used with care—if the `root` password becomes invalid, you may need to reload UNIX.

System Hangs

System "hangs" are most likely to occur when several shells are opened at the same time and then are not closed or exited properly or if the memory buffer for a window (such as the `dg` window) is exceeded with a lengthy file (e.g., a `procpar` file). At that point, nothing on the screen appears to be active—the mouse buttons does not work and commands can not be entered. Some suggestions:

1. Try resetting the console with the reset button inside the console.
2. Try holding down the Stop key on the keyboard (labeled L1 on some keyboards).
3. If another workstation is available and networked with the host, try a remote login (`rlogin`) to the host, identify the hung process (using `ps -ef` in Solaris), and then kill that process.
4. When all else fails, hold down Stop (or L1) and then press the A key.

   Pressing Stop-A should not be used more than necessary because of the chance of corrupting data and files on the disk drive (the CPU is constantly accessing and monitoring the disk drive, and the Stop-A key combination could halt processing while a file was being looked at, thereby damaging the file).

   After pressing Stop-A, wait for the boot prompt to appear, reset the console, and then power up the system.

SCSI Communications Unreliable

Newer software runs faster and as a consequence is more sensitive to hardware that is not running perfectly. If you observe a decrease in reliability of SCSI communications, you can do the following to check the hardware:

1. Check the 5-volt power supply on the acquisition computer backplane. It should be 4.9 to 5 volts and have less than 20 millivolts of ripple.
2. Check the 5-volt power supply on the differential driver box. It should also be 4.9 to 5 volts and have less than 20 millivolts of ripple.

   If either 1 or 2 appears to have low voltage or excessive ripple, you should contact Varian Service.
3. Add an active SCSI bus terminator (Part No. 01-902819-00).
4. Add a second SCSI adaptor board to your computer (Part No. 00-993756-00).

Changing the Name of a Solaris Host

If you need to change the hostname of a computer using Solaris software, the easiest way to proceed is to enter the command `/usr/sbin/sys-unconfig`. Rebooting after this then asks all the questions that were asked at startup about the hostname, IP address, etc. This process removes the `hosts` file, so you should have a copy or be prepared to enter all the information again.

Ethernet Network Troubleshooting

Troubleshooting problems on the Ethernet network involves determining whether the problem lies within the hardware or the software. Try to check for the most obvious problems first (such as cables properly connected and boards properly seated) and then proceed to eliminate factors indicated by suspicious symptoms.

Error Messages

“ie no carrier”
“ie cable problem, Requesting Internet address for 8:20:0:n:n:n”
“Connection timed out”
“? possible cable problem”

Check that the transceiver cable is properly connected to the machine.

Check that the Ethernet Controller board (if applicable) is properly seated.

Check the connections at the multiplexer box (if applicable).

Run the `netstat` command to get packet collision information (see “Commands for troubleshooting the network” below).

“ie ethernet jammed”
“Ethernet transmission error”

Check that the Ethernet cable is terminated at both ends with a 50-ohm terminator.

“machine_name not in hosts database”

Check the `/etc/hosts` file to see if the system name is properly listed along with its complete Internet Protocol (IP) address.

Check that the “Network Information Services” (NIS) database has been updated on the Master NIS file server system and that the associated processes in */etc* are running. To find the name of your NIS server, enter `ypwhich`. To find your nodename in the NIS database, enter `ypcat hosts | grep 'name –n'`.

Find the host system by running the `ping` command (see “Commands for troubleshooting the network” below).

Check the connection with the host system by running the `rlogin` command (see “Troubleshooting commands” below).

“x connection OK - but possible file problem”

Check the `/etc/hosts` file to see if the system name is properly listed along with its complete Internet Protocol (IP) address.

Check `/etc/nd.local` file for proper root and swap entries.

Check `/etc/ethers` file for correct Ethernet address of client machines.

No message or solutions above do not work

Check the `/etc/rc.boot` file for the proper `/etc/ifconfig` statements (without comments). An extra line is needed for a second Ethernet controller on a “gateway” machine.

On Solaris, check `/etc/defaultdomain` and `/var/yp/binding`.

Check that the proper Ethernet controller exists in the system kernel configuration file.

Swap transceiver boxes to determine if the existing boxes might be defective.

Commands for Troubleshooting the Network

The following commands for troubleshooting over the network can be executed when logged in as `root` ("#" prompt), as any VNMR user (">" prompt), or as any other user ("%" prompt).

- To find the host system, enter the `ping` command with the name of the host system you want to check replacing the `host` argument to `ping` (from Solaris only):
  ```
  % /usr/sbin/ping <host>
  ```
  If the program does not find the host system, it displays the message:
  ```
  unknown host <host>
  ```
  For more information, refer to the Sun manual *Administration* (it may refer to the host system as the "machine_name").
- To get packet collision information, use the `netstat` command:
  ```
  % netstat -i
  ```
  A large number of collisions might indicate some kind of cable problem. Refer to Sun manual *Administration*.
- Check the connection by entering the `rlogin` command with the name of the host system you want to join replacing the `host` argument:
  ```
  % rlogin <host>
  ```
  If the connection is not completed, the system might return one of the following error messages:
  ```
  connection refused
  <host>: unknown host
  ```

Console Problems

On UNITY*INOVA* systems, enter the UNIX command `showconsole` to display the console hardware configuration parameters and system versions. Parameter and system version information, which is recorded when the console is booted up, represents the system hardware options recognized by the acquisition computer. `showconsole` is used mainly during troubleshooting and diagnostics.

System Failure

Perform the following corrective actions in case of a system failure. *In the event of an emergency, contact your system administrator.*

Basic Corrective Actions

- *Check spectral window* – Is the full spectrum displayed, or off-center? Check `tof`, `sw`, `np`, `fn`, enter `ds` and click on the Full button. Enter `full`.
- *Check pulse parameters* – Enter `dps` to display pulse sequence. Are the values reasonable? Check `pw`, `p1`, `tpwr`, `gain`, `nt`, `ss`, and `d1`. Which pulse sequence are you running? Is your pulse sequence compiled?
- *Standard parameters* – If you are having problems with parameters, retrieve the parameters or pulse sequence from a standard source.
- *Cancel command* – Is a command running that has not finished yet? Wait for the command to finish, or click on the Cancel Command button. If the command is a

macro, which version of the macro (`vnmrsys/maclib` or `/vnmr/maclib`) are you using?

- *Abort acquisition* – Is an acquisition running in the experiment? To abort the acquisition, enter `aa`. Are there other experiments queued up? Enter `abortallacqs` to abort all acquisitions.
- *Check the probe* – Is the probe tuned? Is the pulse length calibrated?
- *Check hardware* – Are the cables configured correctly? Is the sample seated correctly in the probe? Is the probe seated correctly? Is the VT unit regulating? Is the sample spinning? Is the sample shimmed?
- *Check hardware configuration* – Enter `config`. Check the amplifier types and gradient types.

Advanced Corrective Actions

- *Restart acqproc* – If the acquisition status window shows the Status as Inactive and not Idle, `acqproc` needs to be restarted. In a shell window, enter `su acqproc`. If the message says "Acquisition Terminated," enter `su acqproc` again. Or become `root`, enter `cd /vnmr/acqbin`, then enter `./killacqproc` or `./startacqproc`, as appropriate.
- *Exit VNMR* – If you are having persistent VNMR command problems, exit VNMR (click on Exit VNMR or enter `exit` from VNMR). Restart VNMR (select VNMR from the right mouse button drop-down menu). How much disk space do you have?
- *Log out* – If you are having persistent Sun or UNIX command problems, log out, and log back in.
- *RESET button* – If the acquisition has problems, open the console housing door and press the small red RESET button. You do not need to exit VNMR before performing this action.
- *Power cycle* – If the acquisition cannot be restarted, power cycle the spectrometer by switching off the main power supply. Wait one minute for all devices to stop. Then switch on the main power supply. Wait for the system to come back up. If you power cycle the computer and peripheral devices, be sure to switch them on and off in the correct order.

Color Flashing Problem with Adobe Acrobat Reader

Color flashing can occur if *VNMR Online* (`vnmr_ihelp`) is started before VNMR. Adobe Acrobat Reader, by default, allocates 135 colors in the color map, which does not leave enough colors for VNMR.

To prevent colormap problems, edit the file `/vnmr/acrobat/sol/sparcsolaris/app-defaults/Acroread` by inserting the following line:

```
*targetColorCube:3
```

This line instructs Acrobat Reader to reduce the default number of colormap entries to 35.

Chapter 12. File Security, Repair, and Archiving

Sections in this chapter:

- 12.1 “fsck Command,” this page
- 12.2 “ufsdump Command,” page 176
- 12.3 “ufsrestore Command,” page 178
- 12.4 “tar Backups,” page 179
- 12.5 “Backups Using cron,” page 179
- 12.6 “Data Compression,” page 179

Certain non-standard operations can result in damage to the file systems that cannot be repaired, especially when the system was just about to write a file to disk. Causes include:

- Sudden power failure (e.g., switching off the power on a running UNIX system)
- Interrupting operation (e.g. pressing Stop-a or pulling cables)
- Hardware problem
- Booting the system without an automatic file system check

To prevent permanent loss of valuable data, it is essential to have backup copies of all data on magnetic tape. The optimum way of making tape backups is different for the various file systems—it basically depends on how often and how much each file system is changed.

Tape backups are time-consuming. A full backup of a complete disk on streaming magnetic tape can easily block or slow down your system for an afternoon. But this should not be taken as an excuse for not securing the data. Certainly, the backup time has to be balanced against productive time on the system and against the importance of the data on the disk.

Simply recovering from scratch, reformatting the disk and loading all the software from the original tapes may take about 4 hours, and this does not account for data loss and the time it takes to create the data again. The UNIX tool `cron` can be applied to make backups at times of low computer activities (see “Backups Using cron,” page 179).

12.1 fsck Command

Before making backup copies using the `ufsdump` or `dd` command or after a file system damage, checking the file system with `fsck` is strongly recommended. The command `fsck`, which is usually called automatically during a UNIX bootup, scans a entire file system, or even all file systems, for completeness and consistency. `fsck` tries to repair possible inconsistencies. If it cannot make a repair, it notifies the operator and asks for possible actions, such as deleting or clearing the files that are damaged. Only the superuser should run `fsck`, ideally in the single-user mode when no one else is using the system.

In the multiuser mode, there is an additional problem in that some parts of the file system might have a duplicate in memory. After running `fsck`, any `sync` (which happens periodically due to the `update` daemon and also happens upon operator call or when shutting down the system) will revert any modification in the file system. If after running `fsck`, the message “`FILE SYSTEM HAS BEEN MODIFIED`” shows up, do not use `sync` or `shutdown`, but instead kill the `fsflush` daemon, run `fsck` again, and when `fsck` completes, stop the system by pressing Stop-a (or L1-a). To kill the `fsflush` daemon, find the process-id (enter the command `ps -ef | grep fsflush` and note the PID on the second row of the output line), and enter `kill -9 ###`, where ### is the value of the PID.

While running `fsck`, some actions, such as answering “`y`” to the prompt “`CLEAR?`”, can result in new errors. Therefore, some people suggest running `fsck` twice if any errors are found—in the first run, all prompts except “`SALVAGE?`” and “`RECONNECT?`” are answered with “`n`” in order to find out about the number of errors involved. If the number of errors is high, the original super block is probably damaged. In that case, you should try to replace the super block by one of the backups.

The first backup is in block 32 in each file system. Further block locations can be displayed by the command

```
newfs -Nv /dev/rdsk/c0t3d0s7
```

having the partition `c0t3d0s7`. The replacement is done by entering

```
fsck -F ufs -o b=32 /dev/rdsk/c0t3d0s7
```

Then try `fsck -n` again. Hopefully, the number of errors is decreased.

All you can do in the second run is to answer “`y`” to all questions, forcing the file system to be consistent again. Certain files may be lost completely, while others may be found under a different name in the `lost+found` directory of the corresponding file system.

Whenever `fsck` has modified a file system, check the file system once more to make sure that it is permanently consistent. Normally, this is done automatically during the reboot after `fsck`.

The following examples illustrate `fsck` command syntax:

`fsck /dev/rdsk/c0t0d0s0`	Standard form for interactively checking a single partition
`fsck -n /dev/rdsk/c0t0d0s0`	Run `fsck` with all questions automatically answered “no”
`fsck -y /dev/rdsk/c0t0d0s0`	Run `fsck` with all questions automatically answered “yes”
`fsck`	Interactively check all file systems (as listed in `/etc/mnttab`)

12.2 ufsdump Command

For disk partitions or file systems that are less subject to changes, such as the `/usr` partition, the `ufsdump` command is the best choice for creating backups. In case of symbolic links, the `ufsdump` command copies only the link not the data of the link target (different from the `cp` command).

`ufsdump` is also the most efficient tape command, allowing backing up tapes to their full capacity. By automatically prompting the user for a new tape, `ufsdump` allows dumping large disk partitions onto multiple tapes. A level 0 dump is recommended immediately after installing the complete software. After that, perform the less time-consuming level 1 dumps about once a week, or even less often for the `/usr` partition. Level 1 dumps copy only files that are new or have changed since the last level 0 dump.

Dumps should be made in single-user mode, or after unmounting the partition, or at least when no other operator is using the system. Otherwise, some files might change during the dump, which can lead to possible inconsistencies. It is also recommended switching between two sets of tapes for each dump level.

For a level 0 dump of `/dev/dsk/c0t3d0s5` partition, use the command:

```
ufsdump 0ucbfs 40 /dev/rmt/0 590 /dev/dsk/c0t3d0s5
```

For a level 1 dump of `/dev/dsk/c0t3d0s5`, use

```
ufsdump 1ucbfs 40 /dev/rmt/0 590 /dev/dsk/c0t3d0s5
```

In both commands, the `40` is the blocking factor (option `b`), option `c` means cartridge tape and the `590` is the usable tape length (option `s`), in feet. The usable tape length is usually calculated as 0.95 times the real length of the tape, in feet. Table 11 gives guidelines for usable tape length.

Table 11. Usable Tape Lengths of Tape Cartridges

Tape cartridge	*Real tape length*	*Usable tape length*
QIC, 150 MB, DC-6150	620 ft	590 ft
QIC, 2.5 GB, DC-9250	900 ft	860 ft
4 mm DAT, 5 GB	90 m	280 ft
4 mm DDS-2, 8 GB	120 m	375 ft
8 mm Exabyte, 2.5 / 5 / 10 GB	112 m	350 ft
8 mm Exabyte, 14 GB	160 m	500 ft

When using standard media, both the blocking factor and the usable tape length don't need to be specified. A suitable, time-saving blocking factor is automatically used with Sun tape drives, and `ufsdump` knows about the length of standard media. This allows simplifying the two commands above to

```
ufsdump 0ucf /dev/rmt/0 /dev/dsk/c0t3d0s5
ufsdump 1ucf /dev/rmt/0 /dev/dsk/c0t3d0s5
```

The `0` option specifies the full backup (level 0). The `u` option updates the log file `/etc/dumpdates`, which is necessary for a later restoring. The `f` option specifies the target device or file.

The user partition `/usr` is virtually read-only and, in fact, is seldom altered after installing the complete software. A single backup of that partition should do for the entire lifetime of the software. Frequently used partitions should be dumped by the scheme explained in detail at the end of the Sun manual for `ufsdump` (enter `man ufsdump` in a UNIX shell). Table 12 repeats this dump scheme.

Table 12. Dump Scheme for Frequently Used Disk Partitions

	Sun	*Mon*	*Tue*	*Wed*	*Thu*	*Fri*
Week 1	0 (full)	5A	5B	5A	5B	3A
Week 2		5A	5B	5A	5B	3B
Week 3		5A	5B	5A	5B	3A
Week 4		5A	5B	5A	5B	3B

The numbers 0, 3, and 5 in this table refer to `ufsdump` levels. Such a dump scheme guarantees that any lost file can be recovered from the last day's backup, while minimizing the dump time:

- At the start of every four-week period, a full file system dump is made.
- The level 3 dumps on Friday record all changes since the last level 0 dump.
- The level 5 dumps on the other weekdays only dump files that have changed since the last lower level dump (level 0 for the first week, level 3 for weeks 2, 3, and 4).

This scheme maximizes archiving security while minimizing the dump time. The frequency of the backups can of course be adjusted, depending on how heavily a system is used, and on how important data security is regarded, but it is strongly recommended to stick to a fixed scheme; otherwise, doing the backups is easily forgotten.

The characters A and B in the table indicate tape sets. For optimum security, switching between two sets of tapes for each dump level is recommended, for the case where something happens *during* a backup. A simplified version of this scheme would be, for instance, to do only level 0 dumps on alternating tape sets every weekend.

Dumping the root partition (/) may not be so useful. On one hand, you are working with this partition while dumping it. `ufsdump` keeps a record of dumped files in the root partition, and files for the `ufsrestore` command could be damaged in case of problems with the root partition. A better solution here is to make a backup of files that have changed since installation. To find customized files in `/etc` (such as hosts, printer administration files, etc.) and to save them on a tape, enter:

```
su root
cd /
tar cvf /dev/rmt/0 ‘find etc -mtime -days -a -type f -print‘
```

The command in back quotes is executed first. *`days`* must be replaced by the number of days since the installation. Don't forget to also save customized files such as `.rhosts`, `.profile`, `.login`, and so on. These file names can be appended to the above command. In case of a damaged root partition, you can reload UNIX from the original CD-ROM, preserving all partitions other than root, and reload all files above at the end.

12.3 ufsrestore Command

If file system errors need to be corrected, it is recommended not to restore the whole partition, but to selectively recover damaged files. To avoid causing additional damage, it is best to restore the files first into a temporary directory such as `/tmp`:

```
cd /tmp
ufsrestore xvbf 40 /dev/rmt/0 etc/inet/hosts
```

The last argument is always a relative path, because `ufsdump` stores the files relative to the mount point.

```
cp /tmp/etc/inet/hosts /etc/inet/hosts
```

To restore files selectively, using the interactive option is recommended.

```
ufsrestore bfi 40 /dev/rmt/0
```

gives the prompt "`ufsrestore>`". Enter `help` at this prompt to see the interactive command options.

The blocking factor only needs to be specified if a non-standard blocking factor was specified with the `ufsdump` command. If the default blocking factor was used, the above two `ufsrestore` calls can be simplified to

```
ufsrestore xvf /dev/rmt/0 etc/inet/hosts
```

or, for the interactive version:

```
ufsrestore fi /dev/rmt/0
```

With most disk slices you will first need to create a temporary directory, such as `/export/home/tmp`, for restoring individual data files. If you ever want to restore an entire disk slice, you must of course *not* use a temporary directory, otherwise you would duplicate the contents of the entire slice, and you would most likely run out of disk space. It should rarely be necessary to restore an entire disk slice.

12.4 tar Backups

Because files in user home directories are changed often during day-to-day operation, doing manual tape dumps would be inefficient and time-consuming for the superuser. On the other hand, users do not have the permission to use `ufsdump`. So every user will probably automatically use the `tar` program (see "tar Command," page 149) to make individual backups (archives) of his or her NMR data and the other customized files in the user's home directory.

12.5 Backups Using cron

In multiuser environments, frequent backups are highly recommended. The `cron` command can be quite helpful in this case. It executes jobs at a defined time according to a list in `/var/spool/cron/crontabs/user`. Each user, including `root`, has a `crontab` file named `user`. Editing this file directly has no effect. To activate entries, each user enters the `crontab -e` (edit option) command.

Sometimes an uncommon line editor is defined as the standard editor. You can change to the `vi` editor by setting the environment variable `EDITOR` to `vi` (as `root`, use `EDITOR=vi; export EDITOR`).

To make a backup of `/data` (partition `/dev/dsk/c0t3d0s7`) at 4 o'clock in the morning, from Monday through Friday, you (as `root`) have to enter:

```
0 4 * * 1-5 commands
```

where ***`commands`*** are UNIX commands separated by semicolons. Here the string ***`commands`*** is replaced by, for example,

```
cd /; ufsdump 1ucbfs 40 /dev/rmt/0 590 /dev/dsk/c0t3d0s7
```

Messages from commands that usually go to the standard output are mailed to the user, which allows checking for execution and errors. Make sure that a tape drive is connected to the system and a tape of high enough capacity is loaded. Two tapes should be used, at least exchanging them from backup to backup (see above).

12.6 Data Compression

Over the years, the capacity of disks and tapes has grown continuously. At the same time the requirements on disk space have increased and, with the advent of multidimensional NMR, the size of the FID alone can exceed the size of a single 1/4-inch tape of 150 MB. Data compression is a very helpful tool for reducing the size of files and reducing time especially for archiving on tapes.

Often the size of NMR data, especially if multidimensional or if acquired in 32-bit format, can be reduced by over 40%. Of course, the reason that data files on disks aren't compressed in general is that compressing and uncompressing data takes time and requires additional manipulations. Data compression therefore is suitable for mid-term disk storage and especially for archiving data on tape.

What Files Can Be Compressed?

Any plain file can be compressed: data files, text files, and even compiled programs. Directories cannot be compressed directly (discussed below). The amount of compression depends on the internal structure of the file. Table 13 provides some examples.

Table 13. Examples of File Compression.

File	*Kind*	*Before (KB)*	*After (KB)*	*Reduction*
`/vnmr/bin/Vnmr` (Sun-4)	compiled	1696	952	44%
`/vnmr/parlib/dept.par/procpar`	ascii (text)	14	6	57%
`/vnmr/fidlib/fid1d.fid/fid`	binary, 32 bit	128	51	60%
`/vnmr/fidlib/fid2da.fid/fid`	binary, 32 bit	1056	704	33%

The largest reductions can normally be obtained for ASCII text files and compiled programs. Other binary files such as FIDs are sometimes more difficult to compress (particularly if FIDs were acquired in 16-bit (`dp='n'`) mode or if the VNMR `compressfid` command was used). It is possible for a compressed file to be larger than the source file: for example, `/vnmr/fidlib/dept.fid/fid`). In such a case the `compress` command leaves the source file untouched (unless the `-f` option is used).

In general, FID files, together with their parameters and text files, can be reduced by 30% to 40%; therefore, it is worth the effort. Especially for archiving data, the tape capacity can be increased up to about 50%, with 35% an average reduction, and the time for compressing and uncompressing the data is normally shorter than the transfer time to and from the tape.

Note that 4-mm DDS-2 and 8-mm Exabyte tapes in high-density mode use *compression software built into the firmware of the tape drive*. As compressed files usually cannot be compressed any further, it is questionable whether it is worth compressing data files before saving them on such tapes.

Of course, compressed files cannot be used directly. They must be uncompressed first using the `uncompress` command.

Procedures for Compressed Archiving

FID files are actually directory files that contain the FID, a parameter file (`procpar`), and a text file. Because directory files cannot be compressed directly or recursively, each individual subfile in a FID file would have to be compressed and uncompressed. Because this would be tedious, most people normally combine directories into a single disk file using the `tar` command, as follows:

```
tar cf directory.tar directory
```

This `tar` file can now be compressed in one step. The target file name can be selected freely, but for an easier identification a suffix `.tar` is usually added to the file name. After that, the source directory file can be deleted. This step reduces the disk space requirement and can be regarded as a first level of compression.

On what level should the `tar` command be used? There is a temptation to use the above procedure on complex directories containing lots of subdirectories and files, but that does not make the compressed data transparent enough, because subfiles are “invisible” within a `tar` file. Using `tar` on individual FID directories is recommended. The `compress` command adds a further extension .Z to the file name, and the source file-ID is deleted automatically. A typical procedure for compression might be the following:

```
tar cf sample.fid.tar sample.fid
rm -r sample.fid
compress sample.fid.tar
```

This results in a single, compressed disk file `sample.fid.tar.Z`. In case the `compress` command cannot reduce the size of the file, it leaves the file untouched (and not append the .Z extension). If the `-v` option is used with the `compress` command, it reports the percentage of compression.

The advantage of keeping the complete original file name by adding suffixes is that original file names can easily be recognized when looking for specific data. Keeping the suffix `.fid` is also recommended, because that suffix can distinguish compressed FID files from other compressed files.

The procedure for uncompressing such files is as follows:

```
uncompress sample.fid.tar
tar xf sample.fid.tar
rm sample.fid.tar
```

The `.Z` extension can be omitted with the `uncompress` command.

Due to the `tar` command, this procedure temporarily requires almost twice the disk space before the duplicate file is removed, during both compressing and uncompressing. The `compress` command itself temporarily requires space for the full (`tar`) file plus the compressed file. This means that when the disk is 100% full, it is too late for starting to compress data. In such a case, one way is to store the uncompressed file on a tape, delete the source file, read the `tar` file off the tape using `dd` into standard output, and then compress from standard input to create the compressed file directly:

```
tar cvf /dev/rmt/0 sample.fid
rm -r sample.fid
dd if=/dev/rmt/0 | compress > sample.fid.tar.Z
```

Notice that with this procedure `compress` does not generate the file name automatically.

Both the compression and the decompression can be done in a single step if the `-c` option of the `uncompress` command is used:

```
tar cf - sample.fid | compress > sample.fid.tar.Z
rm -r sample.fid
uncompress -c sample.fid.tar.Z | tar xfB -
rm sample.fid.tar.Z
```

The disk space requirement of this method is between the step-by-step method and the method with the tape. Both the original files and the compressed file coexist after the compression and decompression. The compressed archive files generated with this last method are compatible with step-by-step decompression. Selective extraction is possible with both methods.

To obtain a catalog of a compressed `tar` file, enter:

```
uncompress -c sample.fid.tar.Z | tar tfB -
```

Or even simpler, using `zcat` instead of `uncompress -c`:

```
zcat sample.fid.tar.Z | tar xfB - files
```

To selectively and specifically extract a subfile from a compressed `tar` file type

```
zcat sample.fid.tar.Z | tar xfB - files
```

Chapter 13. UNIX System Customization

Sections in this chapter:

- 13.1 "General UNIX Customization," this page
- 13.2 "X Window System Configuration Files," page 186
- 13.3 "CDE Customization Files," page 186
- 13.4 "OpenWindows Configuration Files," page 186
- 13.5 "VNMR and Other Application Configuration Files," page 187

The entire user interface can be customized by editing a series of files, such as `.cshrc` and `.login`, in each user's home directory. This chapter describes these files.

VNMR stores a set of default customization files in `/vnmr/user_templates`, from which they are transferred into the various home directories by the `makeuser` command when installing VNMR or adding new users. Note that most of these customization files have names that start with a dot, hence the name *dot files*. Dot files can not be seen with the `ls` command unless the `-a` or the `-A` option is selected or if you are `root`. For more information on customization files, refer to the Sun manuals.

13.1 General UNIX Customization

The primary dot file for every user is the personal login start-up file `.login`. Apart from that, many UNIX commands and utilities have dedicated customization files, for example, `mail` uses a file `.mailrc` and the C shell (`csh`) uses `.cshrc`. The "`rc`" in some of these file names stand for "remote control".

.login File

The file `.login` is a shell script that runs automatically after every log in. The main task for `.login` is to define environment variables. There are several UNIX global environment variables, such as `path`, `MANPATH`, `PRINTER`, and `TAPE`. Additionally, there are a number of variables used in connection with VNMR, including `vnmruser`, `vnmrsystem`, `vnmreditor`, `memsize`, `vnmrmenu`, `vnmrtext` (in VNMR 6.1), and VNMR-specific path additions (`/vnmr/bin`, `~/bin`). Additional variables are used for the GNU C compiler (such as `GNUDIR`, `GCC_EXEC_PREFIX`, and GNU path additions) and for the execution of Tcl/Tk utilities (`TCL_LIBRARY` and `TK_LIBRARY`).

Next, the `.login` script (it is a C shell script) determines the environment in which it is run. If it is run on the system console (the nongraphical login screen), it starts up the OpenWindows environment (after setting the `DISPLAY` and `graphics` environment variables and adding `/usr/openwin/bin` to the command path)[1]. If the CDE or OpenWindows environment is already started (from the CDE `xdm` graphical login interface), `.login` essentially only sets the `graphics` environment variable (used by

VNMR), and the rest of the startup is done by startup scripts specific to the selected environment.

If run via a remote terminal or on a virtual terminal (remote network login), `.login` prompts the user to specify the terminal type:

- `g` for GraphOn terminal
- `d` for a dumb terminal
- `t` for Tektronics terminal (the user is also prompted for the exact terminal type)
- `s` for Sun terminal (typically used with remote logins from a Sun window when no remote X session is to be started)
- `x` for a remote X Window system server.

In the case of the remote X server, the user is also prompted for the host name of the X server (a numeric IP address can be entered instead). With many X servers (such as most UNIX workstations), running remote X displays requires the user to first enter

```
xhost +
```

or

```
xhost remote_host
```

prior to starting the remote login, in order to allow remote systems to display information on the local screen via X. The `xhost +` option gives this permission to any remote host, the `xhost` *`remote_host`* option is specific for one system only.

For remote terminals, `.login` also sets the `term` and `graphics` environment variables that allow VNMR to run properly on the respective terminals. It is also possible to run VNMR in non-graphical mode if `d` (dumb terminal) is specified. The same can be achieved by entering

```
setenv graphics none
```

prior to starting VNMR using `vn`. This is of interest in the case of very slow connections. All VNMR commands and macros can be run from the command line with no graphical display, but any text output from the text window or from line 3 (and error messages) is shown.

.cshrc File

The file `.cshrc` is for generating command line aliases—such as `h` as the command for `history`, `m` for `more`, `la` for `ls -a`, and `ll` for `ls -al`—and to set certain variables such as the length of the history or the format of the prompt.

The Sun manual *Getting Started with UNIX* lists a large number of items that can be implemented in `.cshrc`, including features that try to increase data safety by enabling the interactive mode for the `rm` and `cp` commands or features that avoid unintended destruction of windows by disabling Ctrl-d. While some of these features may be acceptable for beginners, they can make life hard for others working on the system.

[1] In this case, the system does not automatically log out when the user exits from the OpenWindows environment. But it is possible to change `.login` to log out automatically after OpenWindows software. Just insert the following lines after the call to OpenWindows (`openwin -noauth`):

```
clear
logout
```

`clear` empties the screen, and `logout` logs the user out, leaving a clear screen with the login prompt. This is only helpful if the nongraphical login is used.

Some alias definitions mentioned in the Sun manuals may even interfere with the execution of certain C shell scripts (note, however, that traditionally most shell scripts are Bourne shell scripts). If you would like to use these features, make sure they are valid in the interactive mode only and not when running C shell scripts. It is perfectly possible to have parts of `.cshrc` only executed during shell scripts and other parts only during interactive mode. See the Sun manuals for more information—the version distributed with the standard VNMR software has a global part only.

C shell scripts can also be written such that `.cshrc` is not called— start them with the line `#!/bin/csh -f`

.exrc File

The file `.exrc` is specific to the `ex` and `vi` file editors and can be used to individually customize for any directory some of the variables for the `vi` text editor, for example, enable the line number display flag. This file is not normally used on VNMR systems.

.profile File

The file `.profile` is specific to the Bourne shell. It is used to customize the Bourne shell environment, which in SunOS is normally used only for shell scripts. This file is not normally used on VNMR systems.

.mailrc File

The file `.mailrc` is a configuration file for the mail software. The file can be generated and modified through the `defaultsedit` tool (SunView) or through a separate, window-based tool started from within Mail Tool in Open Look.

The file `.mailrc` contains lines specific to the `mail` command, to the `mailtool` utility, or to both. It contains a series of set statements and defines variables within `mail` and `mailtool`. Some variables have values—such as bell, flash, and interval—that define the number of beeps and flashes when new mail arrives (very useful when using `mailtool`), and the interval (in seconds) at which mailtool checks for new mail (default is 300 seconds).

If the variable `askcc` is set, `mail` and `mailtool` ask for carbon copy (cc) recipients. If `askbcc` is set, `mailtool` also asks for blind carbon copy (Bcc) recipients. See the Sun manuals for more information about mail software.

.indent.pro File

The `.indent.pro` file is a configuration file (profile) for the `indent` command, which serves to standardize the format of C source files (programs). See "How Should C Programs Be Formatted?," page 204. Normally, this file is not placed in the user's home directory, but rather in the directory that contains C source files. `indent,` unfortunately, is not included with standard Solaris 2.x software.

13.2 X Window System Configuration Files

The files `.xinitrc`, `.Xdefaults`, and `.xlogin` are among the dot files used in the X Window system software environment.

.xinitrc File

The file `.xinitrc` is for the Open Look environment only. It installs the Open Look defaults from the `.Xdefaults` file (see below), starts up the Open Look window manager program `olwm`, calls `.openwin-init` (see below), and defines the Open Look screen background (the file `/vnmr/varian.xicon` in standard VNMR).

.Xdefaults File

The file `.Xdefaults` is for the OpenWindows environment only. It contains the default values that define the behavior (outlook, color, fonts, etc.) of the generic X Window system aspects of the user interface. The contents of `.Xdefaults` are installed through the file `.xinitrc` (see above).

.xlogin File

The `.xlogin` shell script is intended for use (explicit calls) with remote X terminals, in case some of the environment variables required for Open Look have not been defined properly (`OPENWINHOME`, `LD_LIBRARY_PATH`, `path`, etc.). `.xlogin` can often be used in shell tools, instead of logging out and in again. This file is not executable (not is it called automatically). It must be called with `source .xlogin`.

13.3 CDE Customization Files

The CDE customization and definition files are all placed in the local directory `~/.dt`. There is usually no need for the user to modify these files directly. The CDE graphical user interface can be defined, modified, and customized directly and interactively. For further information, consult the Sun documentation or the online *AnswerBook*.

Many CDE applications use a configuration file in `/usr/dt/app-defaults`. A user can customize these through a copy in a local `~/app-defaults` directory.

13.4 OpenWindows Configuration Files

Several OpenWindows configuration files are placed in the user's home directory. OpenWindows applications (such as the Open Look Deskset utilities) use configuration files in `/usr/openwin/lib/app-defaults`. A user can customize these through a copy in a local `~/app-defaults` directory.

.openwin-init File

The file `.openwin-init` is for the OpenWindows environment only. It defines special keys for the Open Look and X.11 environment as well as the windows that show up automatically at startup. These windows and tools can be defined in their size, position, color, and font selection, whether they are open or iconic, and so on, similar to the file `.suntools`. See the Sun manuals for more information.

The Open Look background menu also offers the Save Workspace option that is supposed to freeze the current window setup, so that upon restarting Open Look an identical screen layout is obtained. This is for a simple, flat window setup. It cannot reconstruct parent/child relationships as they exist between the calling window and VNMR, or within VNMR itself, and can therefore lead to error messages and a situation where VNMR is not started automatically.

There are two possible solutions to this problem:

- Don't use Save Workspace and modify `~/.openwin-init` by hand if necessary.
- Use Save Workspace and then fix the resulting file `~/.openwin-init` as follows:
  a. Eliminate the following two lines:
  ```
  toolwait VNMR -geometry +0+0
  toolwait VNMR -geometry +0+160
  ```
  b. Add `Vn &` to the end of the line that defines the window from which VnmrX should be started—either the line ending with `-Wl VNMR` (the window labeled "VNMR") or the line that defines the console window (the line with the -C option).

.openwin-menu File

The file `.openwin-menu` is also for the OpenWindows environment only. This file defines the Open Look root menu. Defining stacked menus is possible (see the Sun manuals for information).

All functions are called here through `exec` command. For each submenu, a `DEFAULT` entry can be defined, which is selected if the menu button is released on the submenu title (without actually entering the submenu). The default can be any item from a submenu. If the `END` keyword from a submenu definition is followed by `PIN`, the submenu can be "pinned down" on the screen background—a useful feature that makes it easier to access frequently used submenus. The Open Look root menu as such can be pinned down anyway.

Other Customization Files

In the OpenLook environment, `clock` and the Open Look file manager (`filemgr`) create their rc files (`.clockrc` and `.filemgrrc`). It is advisable to use the `ls` command without `-A` or `-a` option, due to the proliferation of dot files that clutter the home directory.

The file `.wastebasket` is not a configuration file, but a directory for files that are thrown into the wastebasket from within the OpenWindows file manager `filemgr`. Discarded files can be recovered from the `.wastebasket` directory. The CDE file manager places such files in `~/dt/Trash`.

13.5 VNMR and Other Application Configuration Files

In VNMR 5.1 and earlier, the `.Xdefaults` file contained application-specific definitions such as `pulsetool`, `Acqstat`, etc. for VNMR. In newer VNMR releases, these definitions have been moved into application-specific files in `/vnmr/app-defaults` (`app-defaults/Vnmr`, `app-defaults/Acqi`, etc.).

To customize any of the applications that have files in `/vnmr/app-defaults`, any user can create a local directory `~/app-defaults` and then have local, customized copies of any or all of the files in `/vnmr/app-defaults`. For other Solaris (OpenWindows and CDE) applications, there are similar defaults files in `/usr/dt/app-defaults`, and in `/usr/openwin/lib/app-defaults`.

Most changes in either `.Xdefaults` or `~/app-defaults` concern either colors or fonts. For a list of possible fonts under OpenWindows, use the command

```
xlsfonts | more
```

(don't be scared off by the complex X font names—there are some simpler font names further down the output).

Colors can be specified three different ways:

- Specifying the intensities for red, green and blue as three decimal numbers in the range of `0` to `255`, for example:

```
Window.Color.Background:            255 255 255
Window.Color.Foreground:            0 0 127
```

- Specifying a six-digit hexadecimal number with two digits (`00` up to `ff`) per color, for example:

```
OpenWindows.WindowColor:            #e2e2e2
OpenWindows.WorkspaceColor:         #2dc1e5
```

- Specifying color names, such as `white`, `black`, `RoyalBlue`, `SteelBlue`, etc. A list of possible color names is found in `/usr/openwin/lib/rgb.txt`.

Chapter 14. A Quick Guide to C

Sections in this chapter:

- 14.1 “General Program Structure,” page 190
- 14.2 “Constants, Preprocessor Functions,” page 191
- 14.3 “Types,” page 192
- 14.4 “Variables,” page 193
- 14.5 “Operators,” page 195
- 14.6 “Flow Control,” page 197
- 14.7 “Functions,” page 199
- 14.8 “Pointers,” page 200
- 14.9 “Input/Output,” page 201
- 14.10 “Syntax Checking and Compilation,” page 203
- 14.11 “How Should C Programs Be Formatted?,” page 204

It is beyond the scope of this manual to include a complete introduction to the C programming language. The only goal in this chapter is to give enough background in this language for beginners to read and write pulse sequences and simple C programs.

Considerable literature exists on programming languages, and programming in general, that can be consulted for more complex problems. Anything that specifically relates to pulse sequences can be found in the manual *VNMR User Programming*. Although the source code for VNMR can be purchased, there is no specific information on VNMR programming and programs.

The C programming language was written for and first implemented in UNIX (which by itself is written in C) in the early 1970’s by B.W. Kernighan and D.M. Ritchie, who are also the authors of the classic textbook on C. As a low-level language, C allows writing programs close to the machine language level. By using a limited set of instructions and data structures, C is also relatively compact.

On the other hand, C allows writing programs that are as structured and about as easy to read as Pascal programs, although is does not force the programmer into such a well-defined programming style as with Pascal. While Pascal is an excellent programming language for teaching and learning programming, C is a much more flexible tool for the experienced programmer. It might be said that the disadvantage with C is that it does not control the programmer well enough—that’s why C is so flexible!

C programs often compile despite severe bugs in the program. What is usually a bug in other languages might be used legally in C by an advanced programmer. Fortunately, UNIX contains a tool called `lint` that does about as much syntax checking as any Pascal or other high-level language compiler.

The literature list at the end of this manual mentions a few of the many textbooks on C. Most books on UNIX also contain material on C programming as well.

VNMR itself is written in C and compiled under the SunSoft C (Sun C) compiler. If a user wants to modify VNMR, not only does this require the purchase of the VNMR source code kit and license (VNMR source code is not included with the standard VNMR software license) but it also requires purchase or access to the Sun C compiler. As of Solaris 2.1, a C compiler is no longer included with Solaris. It must be purchased separately.

Probably, few users would want to modify VNMR, but many users would like to compile pulse sequences or other small C programs (the user library also contains many stand-alone C programs). To allow users to compile C programs without the price penalty of the Sun C compiler, VNMR for Solaris comes with the GNU C compiler, which is available for free on the Internet. Pulse sequences and user-defined weighting functions are compiled with the GNU C compiler. As there are some slight, subtle differences between the different C compilers, it should be assumed that `seqgen`, `psggen` (and `fixpsg`), as well as `wtgen` *only* work with the GNU C compiler[1].

14.1 General Program Structure

The global structure of a C program looks like this:

```
#include <stdio.h>

main()           /* main function definition */
{
    function_a();
    statement;
  ...
}
function_a();    /* definition of function_a */
{
    ...
}
```

Lines that start with the number sign (#) are processed by the preprocessor, a program that is run before the compilation starts. The library file `stdio.h` contains a set of standard input and output functions and must be included in every C program.

In C, comments are delimited by `/*` and `*/`. Inserting comments is always recommended, especially because sometimes in C a few characters can define a complex function that needs explanation. Without comments, it is sometimes difficult for a programmer to read even his or her own programs.

C programs consists of functions only, one of which must be called `main`, which is where execution of the program starts. Functions in a C program can be defined in any order. The definitions of functions called within the program are identical to the format of the function `main`, except for the function name. The parentheses are necessary in the header line of a function definition, even though there are no arguments. There is no semicolon at the end of the function header line.

[1]. For the same reason, the VNMR source code (as delivered with the VNMR source code kit) can *not* be compiled with the GNU C compiler.

The beginning and the end of a function body are marked with braces (`{` and `}`). It is good programming practice to write corresponding braces at the same indentation level. Although this might lead to lengthy code, it certainly helps avoid simple mistakes.

The format of a C is free. Spaces and new lines can be inserted at any place, but it is strongly recommended to use indentation, such as inside branchings and loops, to show the structure of a program. "How Should C Programs Be Formatted?," page 204, has more information on this topic.

For pulse sequences, the following basic construction is used:

```
#include <standard.h>
pulsesequence()
{
    ...
}
```

The file `standard.h`, which is stored in `/vnmr/psg` and is a collection of other include files from the same directory, contains `stdio.h`. The only function that has to be written is the function `pulsesequence`. Of course, new functions can be defined and called from within the function `pulsesequence`.

14.2 Constants, Preprocessor Functions

C basically does not know constants. Still constants are often helpful; therefore, they can be introduced via the C preprocessor. The line

```
#define LIMIT 100
```

defines a constant `LIMIT` with a value `100`. The preprocessor replaces all occurrences of "`LIMIT`" by "`100`" from the definition onward. Definitions are usually written directly after the `include` preprocessor statements.

Constants are "typeless" because the preprocessor simply replaces one text string by another (the expression *token replacement* is a better description for this feature). Writing constant names in uppercase is recommended, because it helps distinguish them from variable and function names. Note that there is no semicolon after preprocessor statements; otherwise, for example, the preprocessor would replace "`LIMIT`" by "`100;`".

Constants are also used in the pulse sequence generation software. They are defined in the include files (you do not have to specify the following definitions because they are automatically included in the program by the preprocessor):

```
#define MAXSTR 256
#define A 0
#define B 1
#define C 2
...
```

The constant `MAXSTR` is the length of flag strings and status field related parameter strings, for example, the parameter `dm` might have the value `'nnyn'`. The other constants listed here represent indices for status fields (a large number of constants are available and defined for pulse sequences).

Constants can be defined anywhere in a C program. Unlike C function definitions, constants are only valid after their point of definition. Constants can also be undefined:

```
#define DEBUG
...
```

```
#undef DEBUG
```

In this case, the constant `DEBUG` is only defined between the `define` and `undef` preprocessor statements. The constant `DEBUG` in the above example has no value. A possible application of such constants (we should call them flags in this case) is in conditional preprocessor statements, such as

```
#ifdef DEBUG
...
#else
...
#endif
```

This allows performing certain parts of a program only if the constant `DEBUG` is defined. In pulse sequences, the following preprocessor statements are sometimes used:

```
#ifndef LINT
...
#endif
```

The statements between these two lines are only used if the constant `LINT` is *not* defined. `LINT` is defined automatically when the `lint` tool checks the program. This enables the program to exclude some lines from the `lint` checking (see "Syntax Checking and Compilation," page 203).

Constants can also be defined at compile time, with the `-D` option. In the example above, if the program is compiled with the option `-DDEBUG`, certain conditional parts can be included without changing the source code. Such definitions are sometimes used to incorporate temporary or experimental sections in the source code, or more often, to combine source code for different architectures (such as optimized code for specific chip architectures), or to make a source module for different but related programs.

14.3 Types

The C programming language has only a few basic data types. The type definitions depend on the computer architecture and the C compiler. The definitions mentioned below are those used by typical C compilers on Sun workstations:

`char`	8-bit character (0 to 255)
`short`	16-bit integer number (–32,768 to 32,767)
`int`	32-bit integer number (–2,147,483,648 to 2,147,483,647)
`long`	32-bit integer number (–2,147,483,648 to 2,147,483,647)
`float`	32-bit floating point number (approximately ±5.8774572∗10e-38 to 3.402823*10e38, with about 7 significant digits)
`double`	64-bit floating point number (approximately ±2.225∗10e-308 to 1.797*10e308, with about 15.5 significant digits)
`void`	null type used for functions that don't return a value.

Mostly `char`, `int` and `double` are used, especially for writing pulse sequences. Integer types (`short`, `int`, and `long`) can also be declared `unsigned`, which allows only positive values:

`unsigned short`	16-bit unsigned integer number (0 to 65,535)
`unsigned int`	32-bit unsigned integer number (0 to 4,294,967,295)

Strings are defined as arrays of characters, where the length of the array may not be defined (see "Variables," next).

One-dimensional and multidimensional arrays are possible (see "Variables," next). Other types of arrays are not required for pulse sequences. Neither strings nor arrays need to be defined as types; therefore, in many C programs, there is no type declaration.

As an equivalent to the Pascal record type variables, C allows for structures that are declared as `struct`. See books on the C programming language (described in "Annotated Bibliography," page 329) for more information on structures.

Another type, the pointer variable, is very important in C programs (see below).

For pulse sequences, there is also a predefined type `codeint` for real-time variables, which is a 16-bit integer number. This type does not normally show up in pulse sequences because all real-time variables are usually predefined.

14.4 Variables

A schematic sample program that includes different kinds of variable declarations could have the following form:

```
#include <stdio.h>
int i, ok;
main()
{
    int a,b;
    int c = 0;
    ...
    function_a();
    ...
}
function_a()
{
    int d;
    ...
}
```

Global variables (`i` and `ok` in the example above) are normally defined after the `#include` files and any `#define` constant definitions, usually before the first function. All local variables (`a`, `b`, `c`, and `d`) are declared within the function definition. Local variables are *not* known to subfunctions called within the function where they are local (e.g., `a`, `b`, and `c` are not known to the subfunction `function_a`).

Variables are not initialized automatically, but the variable initialization can be done within the variable declaration (as in `int c = 0` in the above example). The GNU C compiler issues warnings if a variable is used without initialization. Note that variable initializations within an `if` statement lead to a compiler warning, because there is a chance that the variable is used without initialization.

C also allows specifying a storage class for variables: external, automatic, static, and register:

- *External variables* are defined as follows:
  `extern int x;`

Such a variable refers to a global variable `x` that has been defined in another source file, which may have been compiled separately, for the same program.

- *Automatic variables* are the default storage class within a function (including the word `automatic` is not needed in declarations). Such variables are created and destroyed after every function call.
- *Static variables* are only allowed within functions:

  ```
  function_a()
  {
      static double r;
      ...
  }
  ```

  Static values are not created and destroyed for each function call, but they keep their value between each call to the same function.
- *Register variables* are used for variables that are often used in a program. The system tries to keep such variables in CPU registers, to allow for faster execution. Only local automatic variables within a function can be made register variables.

Array variables can be defined easily:

`int z[3];`	Defines an array variable `z` consisting of three integers
`double y[4][4];`	Defines a two-dimensional array `y`, consisting of 4 × 4 double-precision floating point numbers

The individual elements of `z` can be addressed by `z[0]`, `z[1]`, and `z[2]`. Note that in C, the index count starts at 0. Elements of a multidimensional array can be addressed by `y[i][j]`.

Array variables can be initialized with the declaration:

```
int z[3] = {3, 4, 5};
int x[2][3] = {{3, 3, 3}, {4, 4, 4}};
```

String variables are a special kind of array variable:

```
char s[64];
```

This defines a string variable `s`, with a maximum length of 64 characters. String variables can also be initialized with the declaration:

```
char s[64] = "example string";
char m[64] = "";
```

Strings don't have to be filled. C automatically adds a null character `\0` to the end of the string. The null string `""` implicitly contains just the null character. Strings are enclosed in double quotes (`"..."`), whereas character values are enclosed in single quotes (`'...'`):

```
s[0] = 'e';
```

This fills the character `'e'` into the first position of the string `s`. Note that this may lead to a string that is not terminated with a null character. If you are filling a string directly, character by character, you must also fill in the null character, e.g.:

```
s[1] = '\0';
```

The backslash can be used to define any character, especially if nonalphanumeric:

```
char c = '\12';
```

This defines a `char` variable `c` that is initialized with a character of decimal value `12`.

For pulse sequences, the variable definitions are listed in the file `acqparms.h` in `/vnmr/psg`. The real-time variables are defined as (initialized) external variables:

```
extern codeint  ct, oph, ssval, ssctr, bsval, bsctr,   \
```

```
   zero, one, two, three, v1, v2, ... v14;
```

Other variables, especially new variables specific to a pulse sequence, need to be defined and initialized locally. The C compiler reminds you of variables that have been created a second time, and it aborts if there are undefined variables.

14.5 Operators

The *assignment operator* in C is written with a single equals sign:

```
a = 2.3;
```

A number of standard mathematical operators are available:

`+`	addition
`-`	subtraction
`*`	multiplication
`/`	division
`%`	modulus (works only on integers)

Convenient increment and decrement operators for `char`, `short`, `int` and `long` type variables are available:

```
a++
a--
++a
--a
```

There is a distinct difference between the two forms available.

```
a = ++b - c;
```

increments `b` before it is used in the assignment to `a`. The equivalent conventional operations would be:

```
b = b + 1;
a = b - c;
```

If the incrementation operator follows the variable name, it is applied after being used in the statement:

```
a = b++ - c;
```

corresponds to

```
a = b - c;
b = b + 1;
```

Short forms exist for assignments where a variable shows up on both sides of the equals sign:

Short form	*Long form*
`a += c;`	`a = a + c;`
`a -= c;`	`a = a - c;`
`a *= c;`	`a = a * c;`
`a /= c;`	`a = a / c;`
`a %= c;`	`a = a % c;`

The purpose of these short forms is not just elegance: the short form causes the system to load the variable `a` only once into the CPU registers. This has a direct influence on the execution speed of the program.

The assignment statement works like a function that returns the assigned value as result, therefore:

```
a = b = c = d = e + f;
```

is a perfectly legal statement that assigns the same value to the variables `a`, `b`, `c`, and `d`. This works because the assignment operators are evaluated from right to left, whereas the operators `+`, `-`, `*`, `/`, and `%` are evaluated from left to right.

Multiplication, division, and modulus functions all have the same priority and are executed before addition and subtraction. Parentheses can be used to change priorities:

```
a = (b + c) * d;
```

Logical and relational operators are used for boolean operations:

`==`	equality (in Pascal: =)
`!=`	inequality (in Pascal: <>)
`>`	more than
`>=`	more than or equal
`<`	less than
`<=`	less than or equal
`||`	logical `or`
`&&`	logical and
`!`	logical negation

Negation has highest priority, followed by `>`, `>=`, `<`, and `<=`. Equality and inequality are another level lower, followed by the logical `and`. The logical `or` has the lowest priority. To change or clarify priorities, use parentheses.

A complete set of bit operators is also available (see the C programming literature for more information):

`&`	bitwise and
`|`	bitwise or
`~`	binary complement
`<<` *n*	bitwise left-shift by *n* positions
`>>` *n*	bitwise right-shift by *n* positions

C provides for *automatic type conversion* if the two arguments of a binary operator are not of the same type. This conversion happens always upwards: `char` and `short` are converted to `int`, or if one operand is `double`, the other is also converted to `double`. All floating point operations in C are executed in double precision. Also, the two sides of an assignment do not have to correspond in type—conversions occur automatically.

It is certainly not a good programming practice to rely upon automatic type conversion, especially because it doesn't cover all the cases. For example, type conversion does not occur automatically in function arguments. Therefore, type conversions can also be forced with a (unary) type *casting operator*, such as

`(double)`	converts following expression into a `double`
`(long)`	converts following expression into a `long` integer
`(void)`	converts following expression into `void` (nothing)

For example:

```
a((double) n);
```

forces the variable `n` to be converted to `double` before being passed to the function `a`. In fact, many programming errors (often the most fatal ones) are due to type incompatibilities. It is strongly recommended to not mix types. Statements such as

```
double delta, j = getval("j");
delta = 1/(2*j);
```

should be regarded as bad programming practice and should be strictly avoided.

The `(void)` casting operator is sometimes used to discard function return values. Note that casting to integers truncates the value (beware of negative values) and does *not* perform a proper rounding. An example for the use of casting operators, taken from a pulse sequence (simplified):

```
int t1_counter;
t1_counter = (int) (d2 * sw1 + 0.5);
initval((double) (2 * (t1_counter % 2)), v1);
```

14.6 Flow Control

A series of statements can be joined together in a block using braces (`{` and `}`). Such a block acts like a single statement. Within a block, statements are separated by semicolons (`;`), and the last statement in a block is also followed by a semicolon.

Branching

Branchings can be generated by using the `if` statement with the following structure:

```
if (boolean_expression)
    statement or block
else
    statement or block
```

Note that the boolean expression must be written using parentheses:

```
if (a > b)
    a = b;
else
    b = a;
```

or for compound statements after `if` and `else`:

```
if (a > b)
{
    ...
}
else
{
    ...
}
```

Multiple `if` statements can be combined using the `else if` statement:

```
if (boolean_expression)
   statement or block
else if (boolean_expression)
   statement or block
```

```
else if (boolean_expression)
    statement or block
else
    statement or block
```

Instead of a long series of `if` and `else if` branchings[2], there is an equivalent to the Pascal `case` statement: the `switch` statement, which is even more powerful although rather complex in syntax:

```
switch (variable or function)
{
    case value1 : <statement;> <break>;
    case value2 : <statement;> <break>;
    case value3 : <statement;> <break>;
    ...
    <default: statement; <break>; >
}
```

The parts within angle brackets (< and >) are not required. The `switch` statement compares the value of the variable or function given in the first line with the values specified after the `case` keyword. All statements after a matched value are executed until the `switch` statement is terminated or until a `break` is found. Therefore, unless `break` is used, all statements from the match downwards are executed for any matched value. Unmatched values can be collected in the `default` case. The last break is not required, but is often left in so it is not forgotten if another case is added later.

Looping

Looping can be achieved by various statements:

- A loop with a fixed number of passes, the `for` loop
- A loop with at least one pass, the `do` loop
- A loop that starts with a test, the `while` loop.

The `while` loop has the following general form:

```
while (boolean_expression)
    statement or block
```

The `do` loop is used when a loop should be executed at least once:

```
do
    statement or block
while (boolean_expression);
```

Note the semicolon after `boolean_expression`. The `boolean_expression` is true until the loop is terminated[3].

The `for` loop is distinctly different from the corresponding construct in other languages, such as Pascal:

```
for (<statement> ; <boolean_expression> ; <statement>)
    statement or block
```

2. Long chains of `if` ... `else if` ... `else if` ... not only clutter C code and more difficult to read. They are also inefficient in the execution. Furthermore, some compilers may limit the number of sequential `else if` constructs.

3. In Pascal, the boolean expression (in `repeat` ... `until`) is false until the loop is terminated.

The `for` keyword is followed by three expressions in parentheses, for example:

```
for (i = 1; i <= 10; i++)
    statement_or_block
```

- The first expression is normally a statement that initializes the loop variable and is executed before the loop is started.
- The second expression is normally a boolean expression and is run at the start of each loop cycle. If it is false, the for loop is terminated.
- The third expression is a statement that is run at the end of each loop cycle and is usually used to increment (or decrement) the loop variable:

Any of those expressions can be left out but the semicolons must stay there.

If the loop contains a single statement (which can include the loop variable variation), it can also be packed into the third expression (the loop statement would in this case be terminated with a semicolon after the right parenthesis), but this is normally regarded as bad programming style.

Endless loops can be defined as well:

```
for (;;)
    statement_or_block
```

An additional statement `break` allows jumping out of a loop, such as an endless loop:

```
for (;;)
{
    ...
   if (boolean_expression) break;
    ...
}
```

A rather terse form of conditional statements has been used by some pulse sequence programmers, for example:

```
delta = ( (j > 0.0) ? 1.0/(2.0*j) : 0.013);
```

This construct checks if the variable `j` is positive. If it is, the value returned into `delta` is `1.0/(2.0*j)`; otherwise, `delta` assumes a default value of 13 msec. The above construct could be rewritten as

```
if (j > 0.0)
    delta = 1.0/(2.0*j);
else
    delta = 0.013;
```

14.7 Functions

C only knows a single type of subroutine—the function. The structure of a function is outlined in the following example:

```
double calculate(a,b,c)
double a,b;
int c;
{
    int d;
    double e;
    ...
    return(e);
}
```

Like variables, functions are of a certain type. The default function type is `int`, which is often not specified. The value of the function is returned with the `return` statement. Functions that do not return values may be defined as type `void`, for example:

```
void result_message(r)
double r;
{
    printf("The result is: %6.2d \n",r);
}
```

Argument variables are specified by writing the parameter names between the parentheses in the function header line, separated by commas. Argument variables are declared after the header line, on top of the function body (outside the braces), whereas local variables are declared within the braces.

A function is called by its name, as in example above:

```
calculate(x,y,z);
```

Its value can be used in expressions like any variable:

```
result = calculate(x,y,z);
```

Programming languages such as Pascal have two types of arguments: *call by value*, where just the value of the argument is passed to the local parameter, and the *call by reference*, where the value of the variable that is passed (as an argument) can also be changed.

In C, all arguments are always called by value, and the value of a variable given as an argument (`x`, `y`, and `z` in the above examples) does not change. There is, however, a way to change the value of a variable given as an argument in C, by passing pointers to variables, instead of the variables themselves (see below).

14.8 Pointers

Pointers are used very often in C to increase the speed and flexibility of programs. Pointer operations involve the use of two special characters, `*` and `&`:

- If `p` is a pointer variable, `*p` is the value of the object it points to.
- If `v` is a normal variable, `&v` is the pointer to that variable.

Pointers are declared as follows (`int` is the type of the object the pointer points to):

```
int *p;
```

Pointers can be set and used with standard assignment statements:

```
int v;
int *p, *p1, *p2;
p = &v;
p1 = p2;
v = *p1;
```

Pointers play an important role in connection with arrays, where they allow much faster access to individual array elements (as compared to conventional constructions):

```
int a;
int *p;
int array[100];
p = &array[0];
a = *p;
```

```
++p;
```

Pointers can be incremented or decremented to access specific elements of an array; however, C does not check whether the calculated index is inside the array limits.

With functions, pointers can be used as equivalent to a calling by reference. Only values can be passed to a function, not variables themselves, but it is possible to change the value of an external variable instead the pointer to that variable is passed as argument. For example, the call

```
demo(&x, &y)
```

sends pointers to `x` and `y` to the function as follows:

```
demo(a, b)
int *a, *b;
{
    ...
}
```

Pointers are also involved when strings of variable length are used with a single variable:

```
stringfunction(a)
char *a;
{
    ...
}
```

Pointers to strings of any length can now be passed to this function.

To visualize the pointer to an array, there is a second way to define the same function:

```
stringfunction(a)
char a[];
{
    ...
}
```

With an array `a[100]`, the following formulations are equivalent:

```
a is equivalent to &a[0]
a+2 is equivalent to &a[2]
```

In pulse sequence programs, pointers are normally only used for low-level programming.

14.9 Input/Output

C programs can write formatted output:

```
printf("text %d more text \n",a);
```

The `printf` function directs the output to the standard output device, normally the screen or the window, from which the program has been called. The text string is enclosed in double quotes and can include special characters, such as `\n` for the newline character and the formatting strings listed below. For each formatting string (if there are any), an additional variable is supplied as argument. The formatting strings are defined as follows:

%d	decimal number
%u	unsigned decimal number
%o	unsigned octal number
%x	unsigned hexadecimal number

`%e`	exponential notation
`%f`	floating point number
`%g`	exponential notation or floating point number, whichever is shorter
`%c`	single character
`%s`	string

A number after the percent sign indicates the width of the field into which the number (or string) is to be printed:

`%nd`	decimal number, right-justified in a field *n* characters wide
`%-nd`	decimal number, left-justified in a field *n* characters wide

An additional number allows specifying the number of digits after the decimal point:

`%6.2d`	decimal number in field, 6 characters wide, 2 digits after decimal point

Special characters can be printed by using the backslash:

`\n`	newline
`\t`	tab
`\r`	carriage return
`\f`	form feed
`\b`	backspace
`\"`	double quote
`\\`	backslash itself

Input from the default input device (normally the keyboard) can be read by using the `scanf` function, for example:

```
scanf("enter number: %d",&a);
```

This would set the number `a` to a value entered from the keyboard. Unlike the `printf` function, `scanf` requires a pointer to a variable to be supplied as an additional argument. Remember that because arguments are called by value only, using the variable `a` instead of `&a` in the above call to `scanf` would not allow `scanf` to change the value of `a` (and would, in fact, cause a run-time error in the program). The formatting string defines the kind of input expected:

`%d`	decimal number
`%o`	octal number
`%x`	hexadecimal number
`%h`	short integer number
`%c`	single character
`%s`	string
`%f`	real number
`%e`	real number (exponential notation)

The function `fprintf` is used to write formatted output to a file. Within pulse sequences `fprintf` is sometimes used to echo text to the VNMR text window:

```
fprintf(stdout,”textstring\n”);
```

The same formatting strings as for `printf` can be used. Since `printf` also writes output to `stdout`, `fprintf` can be replaced by the simple `printf` call in all pulse sequences.

14.10 Syntax Checking and Compilation

The *C compiler* is called `cc`. It is called by supplying the name of the text (source) file to be compiled. `cc` includes the C preprocessor and a linker for linking after a successful compilation. The source file must have a `.c` extension to its file name:

```
cc sample_program.c
```

By default, this produces an executable output file with the name `a.out`, which already has its execution permission flags set. Therefore, `cc` is normally followed by the command

```
mv a.out sample_program
```

which changes the default name into something sensible. These two steps can be reduced to a single command:

```
cc -o sample_program sample_program.c
```

Programs can be optimized using a built-in code optimizer that operates at different levels (`-O`<*`level`*> option), The lowest optimization level (apart from not using optimization at all) is 1, the highest level is 4; the default level (`-O` option) is 2 (see the UNIX manual page on `cc` for more information about compiler optimizations).

Some examples of optimization are the following:

```
cc -O4 -o sample_program sample_program.c
cc -O -o sample_program sample_program.c
```

Programs that use the `math.h` include library need an instruction for the compiler that tells it to link in the run-time math library (the math functions are not be compiled at runtime, but a precompiled module is linked in at the end):

```
cc -O4 -o sample_program sample_program.c -lm
```

Some programs run faster if the `-fsingle` option is specified with the compilation. C usually converts all values in expressions involving floating point numbers into `double`, before starting the calculation. With the `-fsingle` option, the values remain `float` (as long as no `double` is involved), which speeds up calculations considerably (at the expense of limited mathematical precision (larger round-off errors).

If the GNU C compiler is invoked with the `-DLINT` option, it performs extra, extensive syntax checks:

```
cc -o sample_program sample_program.c -DLINT
```

This often produces a long but useful list of warnings about possible errors.

For pulse sequence programming, the shell script `seqgen` is provided with VNMR, which includes everything from the syntax check and a complete `cc` to renaming the compiled program. `seqgen` can be called from within UNIX (it is located in `/vnmr/bin`) or via the VNMR macro `seqgen`. Both versions accept the sequence name with or without the `.c` extension in the argument. If there is output from the syntax check, an appropriate warning is printed, together with the error messages. Any messages are also stored in a file `sequence_name.errors`.

All sequences from Varian in Palo Alto store the revision number in a static variable `SCCSid` (SCCS is the UNIX Source Code Control System), which is defined and set on the first lines of each sequence:

```
#ifndef LINT
static char SCCSid[] = "@(#)hetcor.c 3.1 Copr 1988 Varian Assoc."
#endif
```

Because this variable is initialized but never used later on (it just allows finding out the source revision number from a compiled module), the syntax check (`lint`) would produce an unwanted error message. This can be avoided by the preprocessor statements on line 1 and 3.

The syntax check sometimes is too accurate and gives warnings for fully correct programs. In pulse sequences, this can happen with multiple `printf` statements that include a variable number of arguments, for example:

```
printf("d3 or jxh must be set!\n");
...
printf("d3 has been set to %5.3f\n",d3);
```

The syntax check produces the message saying that `printf` has been used with a variable number of arguments, which would be a mistake with normal functions. This error message can easily be suppressed through the C preprocessor, exactly as above:

```
printf(”d3 or jxh must be set!\n”);
...
#ifndef LINT
printf(”d3 has been set to %5.3f\n”,d3);
#endif
```

You should, of course, only apply this trick when you are sure that the enclosed source code is correct.

14.11 How Should C Programs Be Formatted?

With the exception of preprocessor statements, which always occupy a full line that starts with a number sign (#), the format of C programs is completely free—spaces, new-line, and tab characters are all treated the same way. Even spaces are only required between character tokens, for example:

```
double f;int i;long ix;
```

It would sometimes even be possible to write an entire C program on a single, long line. In practice, many programmers have their own preferences towards formatting of C programs, to an extent that often makes it difficult for other people to read a program.

Especially for beginners it is strongly recommended to strictly adhere to indentation rules, such as always keeping corresponding braces at the same indentation level. The amount of indentation is debatable: one character is certainly not enough, but for very short C programs like a pulse sequence (20 to 100 lines) two spaces per indentation should be adequate. For long programs, some people recommend 8 spaces per indentation for optimum readability; however, this conflicts with the fact that most terminals, windows, and printers are still 80 characters wide (an old convention), and complex programs with such a indentation would often lead to very long lines. Therefore, an indentation level of 4 should be an acceptable compromise.

Chapter 15. Shell Scripts

Sections in this chapter:

- 15.1 “C Shell Scripts,” this page
- 15.2 “Bourne Shell Scripts,” page 212
- 15.3 “How to Write Shell Scripts,” page 218

Shell scripts are text files containing UNIX commands, mostly interspersed with programming tools (branching, loops etc.), and comment lines. They actually serve as new commands and are executed like commands, by calling them by their name. Solaris 2.x, SunOS 4.x, and most other UNIX implementations have two built-in command interpreters; therefore, two different kinds of scripts exist: Bourne shell scripts and C shell scripts. Each script is covered in this chapter:

15.1 C Shell Scripts

The C shell is the best selection for interactive work (like standard UNIX command entry). It is therefore the default interactive shell. But it is less suitable for elaborate shell script. The programming control structures (e.g., `case` statements) are rather complex, and C shell scripts use more resources and run slower than Bourne shell scripts. For people who know C, however, C shell scripting is probably the easiest to learn, because its syntax is very much like the C programming language (hence its name). For help, enter `man csh` or refer to Sun documentation or any book on BSD 4.2 UNIX.

C Shell Selection, Comments

Irrespective of the calling shell, both command interpreters (Bourne and C shells) are always available (on 4.2BSD UNIX). Therefore, at the beginning of the shell script a selection must be made. The first line in a C shell script must begin with a number, or hash, (#) sign. Also comment lines must start with a number sign.

The most convenient way to write C shell scripts is to start with comment lines (or a descriptive title). This is a good practice and it selects the C shell at the same time. A safer way of defining C shell scripts is to include the following comment line in front of the first command (somewhere in the starting comment, ideally on the first line):

```
#!/bin/csh
```

This calls the C shell command interpreter explicitly. It is possible to speed up the execution with a fast call:

```
#!/bin/csh -f
```

This version does *not* run `~/.cshrc` before starting, which means that the standard aliases are not available. This should be no problem, because shell scripts normally don’t use command aliases.

Set-Variables

Any number of new variables can be generated and initialized with the following statement:

```
set name=value
```

where *name* is any alphanumeric name. The standard convention is that normal variables are written in lowercase and environment variables in uppercase. The value can be a name (no quotes necessary), a numeric value, a string, or a word list (or name list see below). Any reference to that new variable is made by a dollar-sign notation:

```
$name
```

Example for the use of a variable:

```
echo $name
```

A set of standard variables are predefined and globally accessible in the C shell, for example:

`history`	Length of history, e.g., `32`
`prompt`	Prompt for input, e.g., `‘hostname‘-‘who am i‘ #`
`term`	Terminal, e.g., `vt100`
`user`	User name

A list of all so-called *set-variables* can be obtained with the command `set` (no argument).

Environment Variables

Global to both command interpreters is a set of environment variables that are transferred when the two command interpreters call each other. By convention, these variables have uppercase names. A listing of all currently defined environment variables can be obtained with the command `setenv`, for example:

```
vnmr1> setenv
HOME=/export/home/vnmr1
PATH=.:/usr/bin:/export/home/vnmr1/bin:/etc:/vnmr/bin:/usr/c
cs/bin:/vnmr/gnu/bin
LOGNAME=vnmr1
HZ=100
TERM=sun-cmd
TZ=MET
SHELL=/bin/csh
MAIL=/var/mail/vnmr1
PWD=/export/home/vnmr1
USER=vnmr1
vnmruser=/export/home/vnmr1/vnmrsys
vnmrsystem=/vnmr
vnmreditor=vi
memsize=8
vnmrmenu=/vnmr/glide/vnmrmenu
BROWSERDIR=/export/home/vnmr1/ib_initdir
OPENWINHOME=/usr/openwin
LD_LIBRARY_PATH=/usr/openwin/lib
XFILESEARCHPATH=/export/home/vnmr1/%T/%N%S:/vnmr/%T/%N%S
GNUDIR=/vnmr/gnu
```

```
GCC_EXEC_PREFIX=/vnmr/gnu/lib/gcc-lib/sparc-sun-solaris2.3/2
.6.3/
graphics=sun
```

Environment variables can be changed with the same command `setenv`:

`setenv` *`name value`*

For example:

```
setenv TERM sun-cmd
```

Arithmetics with Variables

Simple integer math is possible within the C shell. Arithmetic lines must start with an at symbol (@), followed by a space:

`@ count = 0`	Assign `0` to the variable `count`
`@ count = $num + 3`	
`@ count = $num – 3`	
`@ count = $num * 3`	
`@ count = $num / 3`	
`@ count = $num % 3`	Modulo function
`@ count += 2`	count = count + 2
`@ count -= 2`	count = count - 2
`@ count *= 3`	count = count * 3
`@ count /= 3`	count = count / 3 (truncation of result)
`@ count++`	count = count + 1
`@ count--`	count = count - 1

The operators `+`, `-`, `*`, `/`, and `%` can also be combined to form complex mathematical expressions. Multiplication, division, and modulo operators are executed before addition and subtraction. Parentheses can be used to change the precedence.

A number of bit operators (right-shift, left-shift, complement, logical negation, bitwise exclusive and inclusive OR, and bitwise AND) are also available (see C manuals).

Flow Control

A number of C-like flow controls have been built into the C shell to allow for conditional execution and both conditional and unconditional looping. Conditional execution and looping need a logical function that is either true (numeric value 1) or false (numeric value 0). This is either a `test` function (e.g. for the existence of a file) or a logical expression.

`test` *Function*

The `test` function works like a UNIX command with options and an argument. It can be written in two forms, which are basically equivalent:

`test -f` *`filename`*
`( -f` *`filename`* `)`

For reasons of consistency with C, the second form is usually preferred. The example above tests whether the file `filename` exists and is not a directory. The following options are possible:

`r`	Read access
`w`	Write access
`x`	Executable
`e`	Existence
`o`	Ownership
`z`	Zero length
`f`	Plain file, not directory
`d`	Directory

Numeric comparison operators are also available:

`==`	Equal to
`!=`	Not equal to
`>`	Greater than
`>=`	Greater than or equal to
`<`	Less than
`<=`	Less than or equal to

For example:
```
( $num == 3 )
```

Also, string comparison is possible (do *not* put the pattern in quotation marks):

`==`	Equal to
`!=`	Not equal to
`=~`	Matches (with wildcard characters)
`!~`	Does not match (with wildcard characters)

For example:

`( $str == xyz )`	True if `$str` is `xyz`
`( $str =~ [abc]* )`	True if `$str` starts with `a`, `b`, or `c`
`( $str !~ *.c )`	False if `$str` ends with `.c`

These logical functions can be combined with logical operators:

`&&`	Logical AND
`||`	Logical OR

For example:
```
(( $num == 5 ) && ( -f filename ))
```

The entire expression must be enclosed in parentheses, and again parentheses can be used to group such functions. The logical AND has precedence over the logical OR (and is therefore evaluated first), but for clarity it is recommended to always use parentheses.

Conditional Execution

Conditional execution is accomplished with the `if` statement:
```
if ( <logical_function> ) then
    <commands>
else
```

```
    <commands>
endif
```

Of course, the `else` branch can be left out if it is not required. The `then` must be on the same line as the `if`, and both the `else` and the `endif` should be on separate lines. Multiple `if` statements can also be linked, but in this case there is only one `endif`:

```
if ( <logical_function> ) then
    <commands>
else if ( <logical_function> ) then
    <commands>
else
    <commands>
endif
```

Conditional Looping

Conditional looping is accomplished with the `while` construct:

```
while ( <logical_function> )
    <commands>
end
```

Endless loops can be constructed as follows:

```
while (1)
    <commands>
end
```

Because true has the numeric value 1, this `while` loop executes forever, unless there is a `break` statement inside the loop (see below), or until an external termination (abort) signal is given.

Unconditional Looping

Unconditional looping is used to work through a number of items in a list, using a list variable, for example:

```
foreach suffix(a b c)
    date > time.$suffix
    sleep 3
end
```

This generates the files `time.a`, `time.b`, and `time.c`. The items in the list can also be numbers (rather "number names"). `suffix` is a list variable as it would be set with the following statement:

```
set suffix=(a b c)
```

The individual elements of such a list variable can be accessed as follows:

```
$suffix[2]
```

The numbering starts at 1. The number of list elements (which is 3 in the above example) is accessible as

```
$#suffix
```

A `break` statement allows jumping to the next command after the `end` from within a `while` or a `foreach` loop:

```
while ( <logical_function> )
    <commands>
if ( <logical_function> ) break
```

```
    <commands>
end
<more_commands>
```

Note that there is no `endif` after the `break` command.

case Construction

Multiple `if` statements can be avoided with a C-like `case` construction:

```
switch (<string variable>)
    case <value1>:
        <commands>
        breaksw
    case <value2>:
        <commands>
        breaksw
    <more_cases>
    default:
        <commands>
        breaksw
        <commands>
endsw
```

The `default` branch can be left out if it is not required, and also the last `breaksw` is not necessary, but a good practice, because then it isn't left out when a further `case` is added).

The program flow is the same as in C. Whenever the interpreter finds a match, it executes the commands up to the next `breaksw` or `endsw`.

exit Statement

The `exit` statement can be used to quit the execution of a shell script at any point. It is a good practice to return with some status information:

`exit 0`	Exit with good status
`exit 1`	Exit on error (error status)

Arguments

All arguments given to a shell script command are accessible within the script under the list variable `argv`:

`$#argv`	Number of arguments
`$argv[1]`	First argument
`$argv[2]`	Second argument
`...`	
`$argv[$#argv]`	Last argument

The command `shift` can be used to left-shift all arguments (argument 1 is deleted, `$#argv` is decremented by 1). Shift without an argument operates on `argv`; any other list variable can be given as argument.

The simpler argument syntax using `$1`, `$2`, ... and `$#` from the Bourne shell (see the section "Arguments," page 210) has also been made available within the C shell.

Interactive Input

Instead of setting a variable directly, it can be interactively filled with input from standard input (the terminal), for example:

```
echo -n "please enter filename: "
set name = $<
```

This script regards the entire input line as a single word and puts it into the variable `name`. A way around this is the generation of a list variable:

```
echo -n "please enter filenames: "
set line = $<
set names = ($line)
```

or, in a complex script:

```
echo -n "please enter filenames: "
set line = $<
foreach file ($line)
    <commands>
end
```

Another way to generate a list variable is with the `gets` command, which generates it in one step:

```
echo -n "please enter filenames: "
set names = ‘gets‘
```

Here Documents

Commands that work with standard input (such as `sed`, `awk`, `grep`, and `ed`) can also take their input from the script itself, using a *here document*. The syntax is as follows:

```
command_name << end_marker
lines of data in the here document
end-marker
```

The command often is followed by arguments. The `<<` notation means that it takes its input from the here document. The *`end_marker`* can be any special character that does not appear in the here document, including `+`, `?`, `EOF`, `!`, and `%`. Here is an example:

```
ed filename <<%
g/string1/s//string2/g
w
%
```

This sequence calls the `ed` editor on the file *`filename`* and substitutes all occurrences of *`string1`* by *`string2`*. If sequences with a dollar sign should not be interpreted as variables in the here document, the dollar sign must be escaped with a backslash:

```
ed filename <<%
g/\$s/s//string/g
w
%
```

If substitution should be avoided in the whole document, the end-marker can be quoted out. This is the safer way to avoid substitution:

```
ed filename <<´%´
g/$s/s//$t/g
w
%
```

15.2 Bourne Shell Scripts

Bourne shell scripts have simpler control structures for branching and looping, and they run faster because they use less system resources. Therefore, most longer shell scripts use the Bourne shell. For help, enter `man sh`, check Sun documentation, or look up the topic in almost any book on UNIX.

Bourne Selection, Comments

At the beginning of the shell script, a selection must be made, the same as for the C shell. The lines before the first command call must *not* begin with a number sign (#). A safer way of defining Bourne shell scripts is to include the following comment line in front of the first command, somewhere in the starting comment, ideally on the first line:

```
#!/bin/sh
```

This calls the Bourne shell command interpreter explicitly and allows starting the shell script with comment. Unlike the C shell, there is no fast call for the Bourne shell.

The number sign is used to write comments into the script. Anything after a # sign (up to the end of the line) is regarded as a comment

Variables

Any number of new name variables can be generated and initialized in a simple way:

```
name=value
```

where *name* is any alphanumeric name. The standard convention is that normal variables are written in lowercase and environment variables in uppercase. The value can only be a name (no quotes necessary). Any reference to that new variable is made by dollar sign notation:

```
$name
```

An example for the use of a variable is the following:

```
echo $name
```

There are no real numeric variables; however, variables can be numeric strings.

Environment Variables

C shell environment variables are also known and recognized within a Bourne shell. However, the commands `set` and `setenv` don't exist within the Bourne shell, where such variables are defined just as normal shell variables:

```
GNUDIR=$vnmrsystem/gnu
```

To make such variables known to subshells (this is what distinguishes environment variables from normal shell variables), you must export them:

```
export GNUDIR
```

In VNMR, such environment variables are used to transfer variables (information) from the `seqgen` and `psggen` Bourne shell scripts to their `make` utilities.

Arithmetic with Variables

No direct arithmetic operations are supported directly within the Bourne shell, but the `expr` command can be used to do calculations with numeric strings. This command puts the result of its calculation into standard output. Therefore, to set the value of a variable, you use command substitution. The following examples should be self-explanatory:

`val3=‘expr $val1 + $val2‘`	Add numeric (string) values
`val3=‘expr $val1 - $val2‘`	Subtract numeric (string) variables
`val3=‘expr $val1 \* $val2‘`	Multiply numeric (string) variables (note that `*` has to be escaped by `\*`)
`val3=‘expr $val1 / $val2‘`	Divide numeric variables
`val3=‘expr $val1 % $val2‘`	Take modulo function of a numeric variable

Flow Control

Various flow controls are built into the Bourne shell to allow for conditional execution and both conditional and unconditional looping. Conditional execution and looping need a test function that is either true (numeric value 1) or false (numeric value 0).

`test` Function

The `test` function works like a UNIX command with options and an argument. It can be written in two forms that are basically equivalent (note that the brackets must be surrounded by spaces):

```
test -f filename
```

or

```
[ -f filename ]
```

Each of these examples test whether the file *`filename`* exists and is not a directory. The possible options for test are the following:

`r`	Existence and read access
`w`	Existence and write access
`f`	Existence and plain file, not directory
`d`	Existence and directory
`s`	Existence and size greater than zero

The `test` function can also be used to check the size of strings (name variables), for example:

```
test -z $name
```

This example tests whether the string *`name`* has size zero. The other option is

```
test -n $name
```

This example tests whether the length of the string in the variable *name* is nonzero.

String comparison is also possible:

`test $name1 = $name2`	Check if two strings are equal
`test $name1 != $name2`	Check if two strings are not equal
`test $name1`	Check if a strings is not the null string

For comparison of partial strings, the commands `basename` and `awk` can be useful (see UNIX manuals for more details)—for example:

`base=‘basename $name‘`	Strips off directory part of file name stored in variable `$name`
`base=‘basename $name .c‘`	Strips off directory part of file name stored in variable `$name` and removes suffix `.c` (if present)
`len=‘echo $base | awk '{print length}'‘`	Sets `$len` to length of string `$base`
`name_end=‘echo $base | awk '{print substr($0,length-3,4)}'‘`	Sets `$name_end` to the last four characters of string `$base`

Integer (numeric string) comparison is also available:

`test` *`$num1`* `-eq` *`$num2`*	Equal
`test` *`$num1`* `-ne` *`$num2`*	Not equal
`test` *`$num1`* `-gt` *`$num2`*	Greater than
`test` *`$num1`* `-ge` *`$num2`*	Greater than or equal
`test` *`$num1`* `-lt` *`$num2`*	Less than
`test` *`$num1`* `-le` *`$num2`*	Less than or equal

These functions can be combined with logical operators (for more information about this feature see UNIX manuals):

`-a`	Logical AND
`-o`	Logical OR
`!`	Logical negation

Parentheses can be used to group such functions. The logical AND has precedence over the logical OR (and is therefore evaluated first), but for clarity it is recommended to always use parentheses. Note that parentheses have a specific meaning for the shell and must be escaped with a backslash.

For example,

```
if test $# -ge 3 -a -f $1 -a -d $2 -a -d $3; then ...
```

tests whether there were at least three arguments, whether the first argument is a plain file, and whether arguments 2 and 3 are directories. Only if all four conditions are fulfilled, the statements after the `then` are executed.

The `test` command can also be written in a short form. The last example above then reads as follows:

```
if [ $# -ge 3 -a -f $1 -a -d $2 -a -d $3 ] ; then ...
```

Note: You *must* have spaces inside the brackets.

Parentheses are not required in the above examples, because only logical ANDs are used. An example of a combination of OR and AND is the following:

```
if [ $# -ge 3 -a \( -f $1 -o -d $2 -o -d $3 \) ] ; then ...
```

If you have any doubts about logical precedences between OR and AND, it is better to define them explicitly using parentheses!

Conditional Execution

Conditional execution is accomplished with the `if` statement:

```
if <commands1>
then <commands2>
else <commands3>
fi
```

Of course, the `else` branch can be left out if it is not required. Note that `if`, `then`, `else`, `elif`, and `fi` must be on separate lines (or semicolons must be used to mark the lines). It would, therefore, be possible to write the above sequence as follows:

```
if <commands1>; then <commands2>; else <commands3>; fi
```

Multiple `if` statements can also be linked:

```
if <command1>
then <commands2>
elif <command3>
then <commands4>
else <commands5>
fi
```

Note that in this case there is only one `fi`.

There is also a shorthand version for the `if` statement: && works like a logical AND, and `||` is a logical OR that can be used to link commands:

```
<command1> && <command2>
```

means that *`command2`* is executed only if *`command1`* does not fail. This is the same as:

```
if <command1>
then <command2>
fi
```

For example,

```
[ -f <file1> ] && [ -f <file2> ]
```

On the other hand,

```
<command1> || <command2>
```

means the *`command2`* is only executed when *`command1`* fails.

Looping

Conditional looping is accomplished with the `while` construct:

```
while <commands1>
do <commands2>
done
```

Note again that `while`, `do`, and `done` are on separate lines.

Unconditional looping is used to work through a number of items in a list, for example:

```
for <name> in <value1> <value2> <value3>
do <commands>
done
```

A `for` loop can also be used to scan through the argument variables, see below.

case Construction

Multiple `if` statements can be avoided with a `case` construction:

```
case <variable> in
```

```
    <pattern1>) <commands1> ;;
    <pattern2>) <commands2> ;;
    <pattern3>) <commands3> ;;
    < ... >
    *) <commands4> ;;
esac
```

The patterns can also contain wildcard characters. Only the commands from the first match to the next double-semicolon (`;;`) are executed. Because the wildcard `*` matches all values, it takes the default cases, and the command therein is only be executed if no other match was found. Multiple patterns can be combined, for example:

```
case <variable> in
    <pattern1>|<pattern2>) <commands1> ;;
    *) <commands2> ;;
esac
```

Special character in a pattern—such as a question mark or parentheses—have to be escaped with the backslash (\).

The `exit` statement can be used to quit the execution of a shell script at any point. `I`t is a good practice to return some status information:

`exit 0`	Exit with good status
`exit 1`	Exit on error (error status)

An `exit` with no argument exits with the status of the last command before the `exit`.

Arguments

The command name, the process-ID, and arguments given to a shell script command are accessible within the script:

`$$`	process-ID of the current shell script
`$#`	Number of arguments
`$0`	name under which the script was called (complete path)
`$1`	First argument
`$2`	Second argument
`...`	

The same as C shells, the command `shift` can be used to left-shift all arguments (argument 1 is deleted, the number of arguments, `$#,` is decremented by 1).

The argument variables can be scanned easily with a `for` loop:

```
for i
do >$i
done
```

This script creates an empty file for each of the input arguments. The notation `>filename` creates or clears (if it already exists) a text file. Another example:

```
for i
do case $i in
    -[abc]) <commands1>;;
    -*)     <commands2>;;
    *.[oc]) <commands3>;;
    *)      <commands4>;;
```

```
done
```

In this script:

- *`commands1`* is executed for the options `-a`, `-b`, and `-c`
- *`commands2`* is executed for all other options
- *`commands3`* is executed for arguments that are not options and have a `.o` or `.c` suffix
- *`commands4`* is executed for the rest of the arguments.

Only in this case, where the `in` part of the `for` construction is left, can the `do` stand on the same line as the `for`:

```
for i do >$i; done
```

If the value of a variable should be built into a string, its name must be separated from the rest of the string, for example:

```
tmp=~/bin/data; cat source > ${tmp}1
```

This puts the output of the `cat` command into a file `~/bin/data1`, whereas

```
cat source > $tmp1
```

puts it into a variable `$tmp1`.

If a variable name has not been set, `$name` or `${name}` represents a null string. The braces also allow you to define default values for the case where a variable is not set.

Interactive Input

Instead of setting a variable directly, it can be filled with interactive input from the terminal (actually from standard input, which can also be a pipe). For example:

```
echo -n "please enter filenames: "
read name1 name2 name3 ...
```

This reads a single line from standard input (normally the keyboard). The words of the input line are assigned to the variables *`name1`*, *`name2`*, *`name3`*, etc. sequentially. The last variable takes all the remaining words of the input line. If there are less words than variables, these variables are be set. Variable defaults can be used to define default input:

`${1-'*'}`	Returns `*` if `$1` is not defined
`${2-$1}`	Returns `$1` if `$2` is not defined
`${1='*'}`	Sets `$1` to `*` if `$1` is not defined, then returns `$1`
`${2=$1}`	Sets `$2` equal to `$1` if `$2` is not defined, then returns `$2`.

Here Documents

The basic syntax of *here documents* for Bourne shell scripts is the same as for the C shell (which was described in "Here Documents," page 211), except that the way to quote out the end-marker to avoid substitution in the whole document is accomplished with the backslash in the Bourne shell:

```
ed filename <<\%
g/$s/s//$t/g
w
%
```

Procedures

Shell scripts can be called within shell scripts (with arguments, of course), but apart from that, procedures can be defined within a Bourne shell script. The main purpose of using procedures is to simplify shell scripts by reducing repetitive tasks to procedure calls. Similar to many programming languages, such procedures must be defined before they can be called.The following is a simplified example from a Sun installation script:

```
LOGFILE=sample.log
...
stdin_log() {
    read $1 < /dev/tty
    eval echo "\$$1" >> $LOGFILE
}
...
stdin_log reply1
...
stdin_log reply2
...
```

The main task of this procedure is to capture the keyboard input for a log file (with the `read` command, the keyboard input is not part of the standard output and can therefore not be rerouted into a log file). Note that within the procedure `$1` refers to an argument to the procedure, not to the shell script.

15.3 How to Write Shell Scripts

Any method for generating a text file is adequate for creating shell scripts. Scripts should be placed somewhere in the C shell path; otherwise, the script can only be called with a full (absolute or relative) path. The path is normally set in `.login` (or in `.cshrc`) and includes `/bin`, `/usr/bin`, `/home/vnmr/bin`, `~/bin`, and other command directories. Usually scripts are placed in the personal `bin` file `~/bin`, sometimes in the directory in which they are going to be used (especially if the shell script includes relative pathnames). If a shell script should be made available to all users, it might be placed in `/usr/bin` (change the ownership to `root`). If only the NMR users should have access, it can be placed in `/home/vnmr/bin` (change to ownership to `vnmr1`).

Unless a script is generating by duplicating and then modifying an existing shell script, the new file does not have the execution permission flag set. Enter

```
chmod 755 filename
```

to achieve this (read and execution permission for all users, write permission for the owner only). If only people from the own group should execute the script, enter:

```
chmod ug+x filename
```

Still the system will not find this new command (unless the user has logged out and logged in again) because the C shell reads the command directories only when it is started up and does not know about new additions unless it is notified accordingly. The `rehash` command forces the C shell to update its internal command reference list. (This is not necessary when working with the Bourne shell.)

Chapter 16. **Tcl/Tk**

Tcl/Tk (pronounced “tickle tee-kay”) is used more and more in the VNMR environment. Tcl/Tk is a scripting language and Tcl/Tk tools are special kind of shell script. The scripts are interpreted, not compiled, and much simpler than the equivalent C or C++ programs for the X Window system environment. This leads to some significant advantages:

- It is fairly easy to implement elements of a graphical user interface in Tcl/Tk, which greatly shortens software development cycles. Debugging is very direct, as no compilation is involved.
- The user gets access to the full details of the implementation. Tcl/Tk even allows a user to *modify* these utilities if desired. A typical example for this from VNMR is the `enter` program that queues experiments in automation environments. The program, was explicitly designed to be user-modifiable because the requirements of the various sample changers were far too divergent to be covered in a single user interface.

In VNMR 6.1, quite a number of system utilities, including the dg window, are written in Tcl/Tk. You might want to add your own Tcl/Tk utilities, perhaps for data and system administration and related tasks. For developing new utilities, you definitely should purchase a book on Tcl/Tk, but for just modifying existing tools you should not need to. The section “Specific UNIX Tools,” page 330, in the annotated bibliography lists Tcl/Tk books.

In a future version of this manual, we plan on giving you an outline of the Tcl/Tk language features, so that you don’t always need to refer to the language reference manuals when modifying existing Tcl/Tk scripts. We will also point out some of the special features of the version of Tcl/Tk delivered as part of VNMR, in particular, the windowing shell for Vnmr, `vnmrwish/vnmrWish`.

Chapter 17. Networking via Ethernet

Sections in this chapter:

- 17.1 “Ethernet Configurations,” this page
- 17.2 “Basic Ethernet Network Access,” page 225
- 17.3 “Network Access and Security,” page 228
- 17.4 “Checking the Network,” page 230
- 17.5 “Disabling the Ethernet,” page 231
- 17.6 “Remote Login, Remote Shells, Remote Copy,” page 232
- 17.7 “Remote Printing and Plotting,” page 232
- 17.8 “Advanced Networking: Connecting to the Internet,” page 233
- 17.9 “Mounting File Systems (mount, /etc/dfs/dfstab),” page 234
- 17.10 “Automatic Mounting (/etc/vfstab),” page 238
- 17.11 “Automounter,” page 240
- 17.12 “Using FTP,” page 241
- 17.13 “NIS and NIS+,” page 245
- 17.14 “Heterogeneous Networks,” page 245
- 17.15 “Remote Home Directories,” page 246
- 17.16 “Gateway Machines,” page 247
- 17.17 “Diskless Systems,” page 248

Networking is such a widespread topic with UNIX in general, and especially with Sun computers, that only selected areas are discussed in this chapter. Further details can be found in the sections covering networking in Sun manuals.

17.1 Ethernet Configurations

In the past, an Ethernet network was constructed from 50-ohm cable, which comes in two different qualities: thick Ethernet and Thinnet. Recently, a new standard for low-cost Ethernet connections, twisted-pair Ethernet, has emerged. It provides for Ethernet communication through cheaper cabling and with more flexibility in the network topology.

Standard Ethernet and Thinnet

The basic Ethernet structure consists of a backbone, a linear coaxial cable with 50-ohm impedance that must be terminated with 50-ohm resistors at both ends. Branching in the backbone is strictly forbidden with Ethernet. The maximum length of the backbone depends on the physical layout of the cable.

There are two standards for Ethernet cable: the original, expensive thick cable, which can be extended over 500 m, and the Thinnet or thin Ethernet, which is based on normal 50-ohm BNC cable and can be extended over 180 m. Up to five such cable segments can be linked together with signal repeaters (as recommended by Sun), which results in the maximum total length of 2500 m for thick Ethernet or 900 m for Thinnet, in a single Ethernet network (often called a local-area network, or LAN).

Several such networks can be linked together via gateway stations, which are basically computers with more than one Ethernet interface. Standard Ethernet cable is equipped with high-quality N-type connectors so that shorter pieces can be linked together. In theory, Thinnet and standard Ethernet cables can be linked together, but this is not recommended.

The device that transmits and receives data is called the transceiver (short name for transmitter-receiver). The transceiver consists of the transceiver hardware and of a tap kit that connects the transceiver (normally a small box) with the backbone cable. Note that Varian field service does *not* connect systems into existing networks; this is the responsibility of the user or the network administrator.

Insertion points are marked every 2.5 meters on the standard Ethernet cable. Standard cable lengths should be used for Thinnet. Be sure to use Thinnet cables, not just any BNC cable. Three types of tap kits are in use:

- *N-type inline tap kit* – It has two N-type connectors and can be used as a junction between two standard Ethernet cable segments (not as a repeater). This is the safest connection, but the Ethernet has to be broken for connecting this tap kit, resulting in a temporary communications loss in the entire Ethernet segment.
- *Vampire tap kit* – At one of the insertion marks a hole is drilled into the standard Ethernet cable, which allows connecting a new transceiver without interrupting communication. Special tools are needed.
- *Thin Ethernet or standard BNC tap kit* – Although it has two BNC type connectors, some thin Ethernet tap kits only have a single BNC connector. In such a case, a T junction has to be used immediately at the tap kit (no branching is allowed with Ethernet) and the two segments of the Thinnet must be attached to it. If the tap kit is located at the end of the network, a T junction is still required with connections to the Ethernet, the tap kit and to a 50-ohm resistor. Also here, the backbone has to be broken up for the installation, causing an temporary loss of communication in the entire Ethernet segment, unless there are spare (unused) T-junctions in the Thinnet backbone.

The Ethernet controller is connected to the transceiver via a branch cable, which uses 15-pin connectors and can be up to 50 m in length. A computer connected to Ethernet (computer, controller, transceiver, and tap kit) is also called Ethernet node. Up to 100 nodes are allowed per each 500 m standard Ethernet segment.

Most Sun computers are equipped with an Ethernet controller with branch cable connector, either directly on the CPU board or through a special interface cable., newer models are equipped with a 10/100BaseT connector, see below.

Connecting to the Network

Once the software is installed and set up to run the Ethernet network, the network hardware can be connected. You may perform the installation yourself, but if you have purchased a Point-to-Point Thinnet Network (Varian Part No. 00-992489-00) or an additional Thinnet Node (Varian Part No. 00-992491-00), a Varian service engineer will perform the installation, subject to the following limitations:

- The network consists of Thinnet products listed above.
- The installation takes place at the time of the installation of the spectrometer.

- The computers at both ends of the network were purchased from Varian.
- The cabling can be routed on the floor and does not involve walls or ceilings.
- The connection does not involve connecting to an existing Ethernet network.

Thinnet is a network composed of thin Ethernet coaxial cable (RG/58U). Each end of the cable connects to a transceiver via a BNC T-connector, and the transceiver connects to an Ethernet port on the rear of the computer. A 50-ohm terminator must be placed into any BNC T-connector port not attached to the transceiver or a cable. The configuration of the Point-to-Point Thinnet Network is shown in Figure 22. It consists of two transceivers (called µtransceivers due to their small size), two BNC T-connectors, two 50-ohm terminators, and a length of thin Ethernet.

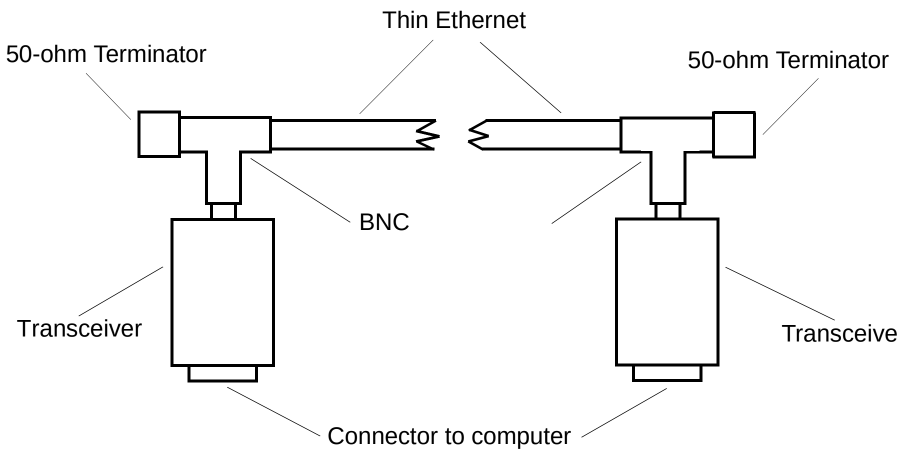


Figure 22. Point-to-Point Thinnet Network

Because of the nature of thick Ethernet cabling and the connections made to it, Varian policy is to not install thick Ethernet. This includes installing the cables, transceivers, and checking the installation. The instrument owner is responsible for correct completion of the installation of a network using thick Ethernet cabling.

If you decide to perform the installation yourself, the following instructions apply.

1. Follow the instructions in the "Normal UNIX Shut Down," page 68 of this manual to shut down the system and power down the workstation.

CAUTION: **Do not install Ethernet cable with power on. Blown fuses can result.**

2. If installing a Thinnet (thin Ethernet) network, connect a transceiver to the Ethernet port on each workstation, attach a BNC T-connector to each transceiver, and connect thin Ethernet cable between the transceivers. Attach a 50-ohm terminator to each BNC connector without a cable.

   If installing thick Ethernet, connect the cable from the workstation to the nearest Ethernet transceiver (from the InLine Tap Kit, Part No. 00-968492-00). Refer to the Sun Microsystems Hardware Installation Manual for the procedure. On some Sun systems, a jumper must be changed.

3. Follow the instructions in the section "Restarting UNIX," page 69 to turn the power on to the workstation and reboot the system.

4. Log in as `root`, enter the `root` password (if implemented), and then confirm network operation running `ping`, where you replace the *`hostname`* argument with the name of your host system:

   ```
   # ping hostname
   ```

   You should get the message below:

   ```
   hostname is alive
   ```

Details of the remote log in operation and other networking features are available by entering `man rlogin` or by referring to the Sun Microsystems documentation.

Disconnecting from the Network

If your system is going to be (or has been) removed from the Ethernet network, enter the `setnoether` command and reboot the system, as follows, before running an acquisition.

1. Log in as `root`, and enter the `root` password (if implemented).
2. Enter the following commands:

   ```
   # cd /vnmr/bin
   # ./setnoether
   ```

3. Reboot the system:

   ```
   # init 6
   ```

The `reboot` command does a shutdown and a boot on Solaris.

If Ethernet is disconnected and `setnoether` is not run, the display in Figure 23 may appear if a performance meter is running on the system. This display indicates the CPU is busy looking for a non-existent Ethernet and cannot do acquisitions, lock, or shim.

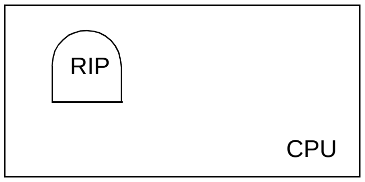


Figure 23. Performance Meter with Nonexistent Ethernet

Twisted-Pair Ethernet

Twisted-pair Ethernet (TPE) uses a completely different topology All systems connect with a central hub in a star-shaped configuration, and several hubs can be connected again, using twisted pair cabling. Converters allow connecting TPE networks with conventional (BNC or Thick Ethernet) networks. The cabling is TPE networks is done with either shielded or unshielded twisted pair cables with RJ-45 connectors (same as used for telephone connections). In NMR laboratories, shielded cables are strongly recommended. TPE networks operate at 10 Mbps (megabits per second) for a 10-MHz base modulation rate.

Recently, 100-Mbps *fast Ethernet* has become available. Fast Ethernet uses the same TPE cabling standard, but high-quality shielded cables specified for 100 Mbps are highly recommended.

Most or all fast Ethernet ports support *both* 10 Mbps (10baseT) *and* 100 Mbps (100baseT) communication, even on the same network. The Ethernet hardware automatically adjusts the transmission speed to that of the current communication partner. The Ultra 1 models with Creator graphics controller (Ultra 1/170E, Ultra 1/200E) and the newer Ultra (Ultra 5, Ultra 10, Ultra 30, Ultra 60)workstations are equipped with a fast 10/100baseT Ethernet port. For earlier workstations, fast Ethernet ports are available through an SBus expansion board.

17.2 Basic Ethernet Network Access

After computers have been linked via Ethernet, their communication software needs to be activated; otherwise, the systems don't know about Ethernet or other computers. The Ethernet communication is a complex multilayer process. In this manual, we only look at those parts of the software that are essential for the user.

The best and safest way to define the network parameters and set up a workstation for network access is not to start manipulating the network definition files by hand, but to either start entering the proper networking parameters upon loading Solaris or (if Solaris is loaded already) to enter

```
sys-unconfig
```

which removes all local information from the networking definition files in `/etc` and reboots the system. Upon booting, the user is then prompted for the necessary information, such as the local host name, the IP address, and netmask (see also the manual *VNMR and Solaris Software Installation*). Other hosts on the network can then be added using the Solaris AdminTool. However, for debugging purposes it is often useful to be able to check the networking information directly in the administration file. Therefore, we provide this information as if you were modifying these files by hand.

Ethernet Hardware Address

The first thing to consider is that each of the systems must have a unique address for identification in the network. Each Ethernet device has a *hardware address*, a PROM-based sequence of hexadecimal numbers that is unique worldwide (license and distribution by Xerox). A hardware address might be, for example:

```
8:0:20:1:6:4a
```

The user does *not* need to know this address on standard systems (it is read out from the PROM when booting up UNIX), unless the limNET program is being used.

Numeric IP Address

More important than the hardware address is the *Internet address*, which is user-definable. The software looks up the Internet address in an internal library `/etc/hosts`. If the Ethernet is not activated, this file, created by the `suninstall` program, consists of a single line (apart from comments):

```
127.0.0.1      unity localhost loghost
```

This defines `unity` as local host and also as loghost. The number `127.0.0.1` is the default Internet address for systems without active Ethernet. It defines `unity` as host number 1 (0.0.1) on network number 127

The Internet address consists of two parts: the host number and the network number. The address has a total length of 4 bytes (with a value of 0 to 255 each). In a simple network, to communicate, *all hosts communicate must use the same network number, and each host must have a unique host number.*

Three classes of Internet addresses exist:

- *Class A* – The first number between 0 and 127. The last 3 bytes define the host address. This allows defining 128 networks with up to 16,777,214 hosts each (x.0.0.1 – x.255.255.254).

- *Class B* – The first number between 128 and 191. The network and the host address. are defined by 2 bytes. This gives 16,384 network numbers (128.0 – 191.255) with up to 65,534 host numbers each (x.x.0.1 – x.x.255.254).
- *Class C* – The first number between 192 and 255. The network number uses 3 bytes but only 1 byte as the host number, resulting in 4,194,304 different network numbers (192.0.0 – 255.255.255) with up to 254 host numbers each (x.x.x.1 – x.x.x.254).

Do not use host numbers that have either all bits set or all bits unset (0x00 and 0xff, or 0 and 255 for a class C address), because either one of the two (depending on the network protocol) is reserved as the *broadcast address*, through which all hosts on a network can be reached.

For a purely local stand-alone network, the choice of the network class and network number is arbitrary. Most people choose to stay with Sun's default class C network number (192.9.200). The choice of host number is completely up to the local network manager (within the restriction mentioned above).

For a linked (internetworked) network (through gateways or through attachment to the Internet network), the network number cannot be freely selected. It must be ensured that the same network number is used on all systems that should be reached directly (without going through a gateway system). If you are hooking up to an existing network, you are given both the host and the network numbers from the network administrator. If you intend to hook a new network onto existing other networks, you have to talk to the networking facilities manager to obtain a new network number or you should obtain an official network number from your Internet service provider (ISP).

On newer spectrometers (UNITY*INOVA*, *MERCURY*-Series, and *GEMINI 2000*) the acquisition computer is connected to the host through Ethernet. On this network, network address 10 is used, a class address officially reserved for internal networks. Therefore, do not select this network address on your local network. The term "internal network" implies that this network will never be visible to the public (e.g., there is no Internet packet routing to the acquisition computer). If you have a local network with the network address 10, you can certainly expect problems when connecting to Internet, because you are not using a network address that is officially assigned to your site.

Netmasks, /etc/netmasks

Larger institutions often posses a class B network number for the entire facility, because there may be far more than 255 hosts. For security, however, it may be undesirable to let everybody talk to everybody else throughout the institution. Also, distributing the network over several subnets, instead of using a single large backbone, reduces the network load. For these reasons, the facility manager may decide to divide the facility network into several subnets that behave like different networks. You may receive a 3-byte class B IP address, such as 135.24.68. For you, this is the same as a class C network number, because byte 3 of the address has been declared part of the network number. The mechanism which allows doing this is called a *netmask*.

The netmask defines which bits within an Internet address define the network number, in cases where a higher class network (typically class B) should be split into a series of subnetworks. The default netmask is 255.0.0.0 for class A networks, 255.255.0.0 for class B networks, and 255.255.255.0 for class C networks.

In the example given above, the netmask is probably 255.255.255.0. This is the most common nondefault netmask, although in theory any number of bits in the host part of an Internet address can be declared part of the network number. For example, a small institution with a class C network could create four subnets (up to 62 hosts each) using a netmask of 255.255.255.192. A network number and the associated netmask are defined in a file `/etc/netmasks` (on systems without subnetting this file can be left untouched). For example, in this `netmasks` file,

```
135.24.6.0   255.255.255.0
```

the network `135.24.6.0` is associated with the netmask `255.255.255.0`:

/etc/hosts File

If you are going public with your network (e.g., through public e-mail access), you need to obtain an IP network address that is *unique worldwide*. In the United Stated, contact a Internet service provider (ISP). For Europe this authority has been delegated to the national branches of EUnet (the European Internet)[1]. You need a good justification (over 255 hosts within 5 years) for a class B network number. Class A addresses are not available for those who don't have one already. It is easier to obtain a class C address.

What does the `/etc/hosts` file look like on a system within a network? Here is an example:

```
192.9.200.1      inova500 loghost
192.9.200.2      unity400
192.9.200.3      inova300
192.9.200.4      sun
127.0.0.1        localhost
```

The list is generally self-explanatory. The address `192.9.200` is the default (class C) network address given by the `suninstall` program. When loading UNIX, this number can be modified. The host number is set on the Workstation page, and the network number on the Defaults page. The `suninstall` program creates a correct file `/etc/hosts`. Other systems, however, have to be added by editing `/etc/hosts` (be sure to make a backup copy of the original file before starting to edit it). Remember, do *not* use the host numbers 0 and 255 for any of the four fields.

Several separated networks (or subnets), even several network classes, can coexist on the same cable segment. Only gateway machines with two Ethernet controllers can access two networks (or more, with more controllers) simultaneously. They also have one host address and one host name per network. The gateway machine is the only possibility for hosts on different networks to talk to each other.

On UNITY*INOVA* spectrometer hosts, `/etc/hosts` also contains standardized entries for the communication with the acquisition console. The network IP address for this is 10, which is a class A address reserved for internal networks (i.e., networks that are not intended for communication with the rest of the world). These spectrometer entries are

```
10.0.0.1       wormhole
10.0.0.2       inova
10.0.0.4       inovaauto
```

In this example, `wormhole` is the name of the spectrometer host (on the acquisition network only), `inova` is the host name for the acquisition CPU, and `inovaauto` is the host name for the Magnet and Sample Regulation (MSR) board. *This makes it obvious why* `inova` *must not be chosen as hostname on spectrometer hosts.*

[1] For Germany, the e-mail address is hostmaster@deins.Informatik.Uni-Dortmund.de; for Switzerland, it is hostmaster@eunet.ch.

On *MERCURY*-Series and *GEMINI 2000* spectrometers, there are two entries for the acquisition in `/etc/hosts`:

```
10.0.0.1       wormhole
10.0.0.21      gemcon
```

/etc/hostname.*, /etc/nodename Files

Apart from `/etc/hosts`, the local host name must also be stored in `/etc/nodename` (this file contains just the local host name, nothing else). The UNIX start-up scripts in `/etc/rc.d` check for a file named `/etc/hostname.*`. On a stand-alone system, this file is named `/etc/hostname.xx0` or `/etc/hostname.lo0` and contains the local host name (same as `/etc/nodename`). If one of these two files is found, the Ethernet hardware is not activated at bootup time.

On networked systems, there is one file `/etc/hostname.*` per network interface (two or more on gateway machines), and it contains the local host name for the corresponding network. The file name extension is the name of the network:

`hostname.le0`	Standard Ethernet, first (standard) port
`hostname.le1`	Standard Ethernet, second port (typically on an SBus card)
`hostname.mce0`	10/100baseT Ethernet, first (standard) port
`hostname.mce1`	10/100baseT Ethernet, second port

The `le` in the above port names stands for Lance Ethernet, named after the manufacturer of the Ethernet interface chip. On some older systems, different (Intel) Ethernet hardware was used, and the corresponding port names then were `ie0`, `ie1`, etc.

17.3 Network Access and Security

On a stand-alone system, the password mechanism provides a reasonable level of security (in connection with the file permissions). In a network, system security becomes more complex:

- It is undesirable to always have to type the password when working across a network (especially in your own account on a remote system).
- Certain remote commands (`rsh` and `rcp`, see "Remote Login, Remote Shells, Remote Copy," page 232) do not work if a password has to be specified.

Global Access Control, /etc/hosts.equiv

The most important file for access control is `/etc/hosts.equiv`. For the above example, a possible `hosts.equiv` file would be the lines:

```
inova500
unity400
inova300
sun
```

This file is a list of *trusted hosts*, the host names of systems allowed to have direct access, such as via `rlogin` or `rsh`, on that system. If a user (the name, not the ID counts) is defined on both systems and does a remote login into the account with the same name, no password is checked if the remote system is listed in `/etc/hosts.equiv.` Remote shells (`rsh`) and remote copying (`rcp`) are only possible if the user or the host is trusted through `hosts.equiv`. Note that `hosts.equiv` is only valid for users other than `root` (for `root`, the file `/.rhosts` provides the same functionality, see below).

In the example above of `hosts.equiv`, every host name is on a single line. The above form would give any user from the listed hosts remote access (the local host name doesn't need to be listed, but including it simplifies the administration because the same file can be used throughout a network). If the file consists of a single plus (`+`) sign (the default after loading SunOS), every host is trusted. Host names that are preceded by a minus sign (`-`) are explicitly not trusted (users from such a host would have to use passwords when executing a remote login, and remote shells or remote copying is not allowed).

If only the host name is listed in `hosts.equiv`, every user from that host is a trusted user. It is also possible to specifically exclude certain users from a given host from the `rsh` and `rcp` commands (for remote logins, these users always have to use a password):

```
inova500 -vnmr2 -vnmr3
```

We can also deny access to the local system to a specific user from *any* host:

```
+ -vnmr2 -vnmr3
```

On the other hand, it is also possible to further open up the access for a remote user:

```
inova500 vnmr1
```

This allows `vnmr1` from `inova500` to do a remote login into any local account (except for `root`) without having to supply a password. Remote shells are also possible using any account except `root`. For `rcp`, the remote user name determines the local write permission. It is obvious that this feature should be used very restrictively (if at all); otherwise, the system security could be severely affected. The same access can also be granted to a specific user from any host:

```
+ vnmr1
```

We could even open up this facility to *every* user from a remote host (again, except for `root`), but this is certainly most undesirable from a security point of view:

```
inova500 +
```

Individual Access Control, ~/.rhosts

Any user can specify additional hosts and users that are allowed (or denied) logging into the user's account without being asked for a password, overriding or complementing the data in `/etc/hosts.equiv`. This is achieved with a file `.rhosts` in the user's home directory (`~/.rhosts`). The functionality and the structure of this file are identical to the file `/etc/hosts.equiv`, except that if users are specified behind a host name, access is only granted to the local account, not into any account, as with `/etc/hosts.equiv`. It is advisable to only specify users in `~/.rhosts` (of course, it is acceptable to deny access for specific users through `/etc/hosts.equiv`).

For security reasons, `root` is a special case. For `root`, only the file `/.rhosts` is checked, not `/etc/hosts.equiv`. While opening up the access to normal users, `root` can stay restrictive in the access into its own account. It is crucial for the network security that `/.rhosts` is set up in a very restrictive way.

It is obvious that the files `/etc/hosts.equiv` and `~/.rhosts` are sensitive points with respect to security, especially on larger networks. It is therefore recommended to carefully check entries in `/etc/hosts.equiv`. Even more important than `hosts.equiv` and `.rhosts` is the password protection, of course. Even the most restrictive use of the files mentioned above is worth nothing if `root` (or any other account) does not have a password on a single machine in the network. It is strongly recommended that the network administrator periodically check all accounts for the presence of a password.

A good password is not a word, an abbreviation, or a combination of characters that can be found in any book, newspaper, or dictionary. Users should be strongly discouraged from using a birthday, a name of a friend, or a pet as password, or character sequences that can be easily decoded when they are typed in. We recommend using a combination of alphabetic and numeric characters, some uppercase and some lowercase. Passwords should be at least six characters long (better eight, which is the maximum number of significant characters in a password). Furthermore, urge users not to write down their password and to change their password periodically, like once a month.

17.4 Checking the Network

Once the Ethernet is turned on (see above), the kernel starts looking for the Ethernet hardware, even if there is no communication going on. This provides some basic and continuous system diagnostics.

- Should the branch cable not be connected (transceiver cannot be reached), the following error message appears in the console window:
  ```
  ie0: no carrier
  ```
  or
  ```
  le0: No carrier - transceiver cable problem?
  ```
  Such messages show up every minute or so, or more often if any attempt to communicate via Ethernet is made. This can also indicate that the branch cable is just loose. Unfortunately, the mechanism that fixes a branch cable to the connector does not quite meet the standards of the other Sun or Ethernet hardware. Be careful not to move the cable once it is installed.
- The following message means that the Ethernet hardware has not been activated (no file `/etc/hostname.le0`)
  ```
  network is unreachable
  ```
- The following messages indicate that the Ethernet transceiver is not hooked up to the backbone cable or that the backbone is not terminated correctly.
  ```
  Ethernet jammed
  ```
  or
  ```
  Ethernet cable problem
  ```

The next step is to check Ethernet with real communication programs. The first question is whether the other host up and running, and is its software (also the local files) for Ethernet set up correctly?

The first test is use the `ping` command to see if the two CPUs recognize each other via Ethernet. The usage of `ping` and the reply, if successful, for a remote host named *`otherhost`*:

```
ping otherhost
otherhost is alive
```

In case of problems, the response would be as follows (after several seconds):

```
ping otherhost
no answer from otherhost
```

The next higher level of testing is use the `spray` command to transmit data and to check how successfully and efficiently the data transfer works. Ethernet sends data in packets. If two devices transmit data at the same time, this results in a collision, and the packets will be dropped and transmitted again, which will affect the transfer rate, of course. Entering the `spray` command with a remote host name might give the following response:

```
spray otherhost
sending 1162 packets of lnth 86 to otherhost ...
        no packets dropped by otherhost
        219 packets/sec, 18855 bytes/sec
```

The number of packets and their length can be varied. The maximum length of a packet is 1514 bytes:

```
spray otherhost -c 1000 -l 1514
sending 1000 packets of lnth 1514 to otherhost ...
        72 packets (7.200%) dropped by otherhost
        133 packets/sec, 202448 bytes/sec
```

Larger packets obviously make the communication much more efficient. In practice, however, the standard test (`spray` with only the host name as argument) is much more critical as check on collisions and other communication problems. `spray` can serve as a test for the maximum practically achievable transfer rate at a given packet size.

If `ping` and `spray` are working, this proves that the Ethernet connection is functioning properly, that both hosts are up, their Ethernet enabled (`/etc/hostname.le0`), and that the file `/etc/hosts` is set up correctly.

A final level in the test procedure is to check programs with remote operations. This requires `/etc/hosts.equiv` to be set up correctly. These commands are `rlogin` (remote login), `rsh` (remote shell) and, and `rcp` (remote copy). Each is described in "Remote Login, Remote Shells, Remote Copy," page 232.

The commands `rlogin` and `rsh` do not ask for a password, as long as a host is listed in `/etc/hosts.equiv` on the remote system and as long as the user name is known on both systems. Users from hosts that are not listed in `/etc/hosts.equiv` need to supply a password.

The command `rcp` only works if no password is requested and if write permission exists in the target directory. If `rsh` works without problems and without password, `rcp` should also work without problems. If `vnmr1` has a home directory on both systems, it should be possible for `vnmr1` to copy between the two home directories and to or from any of the `/vnmr` directories. `root` may have a problem copying into a remote directory. Its user-ID on the remote host is reset to –2 (65534), which is equivalent to "nobody," and therefore `root` can only write into remote directories with global write access. In general, it is better to use remote copy to copy files into the local account (if necessary, do a remote login first).

17.5 Disabling the Ethernet

Just unplugging the Ethernet or the branch cable results in messages that show up periodically in the console window. If any attempts to communicate are made, these error messages accumulate dramatically, which is the least of all possible troubles. In some situations, the computer can be locked up due to continuous attempts to do something over the network. When operating in a nonwindow environment, the error messages actually pop up in the middle of the current command line. When using a full-screen editor like `vi`, it disrupts the screen.

Therefore, it is strongly recommended that Ethernet is properly disabled. This is achieved by renaming the file `/etc/hostname.le0` (or `/etc/hostname.mce0` on some systems) into `/etc/hostname.xx0`. After that, the system must be rebooted completely; otherwise, the Ethernet controller will not be deactivated.

In theory, you could directly deactivate the Ethernet hardware with a command such as `/etc/ifconfig le0 down`, but because rebooting also switches off any daemon that could try to use Ethernet, renaming the file is safer. Consider active disabling only for emergencies such as if the system is flooded with Ethernet hardware error messages.

The next step is to make sure no part of the software tries to use Ethernet. This mostly concerns mounting (see below) and NIS (see "NIS and NIS+," page 245), where additional steps are necessary to shut down these activities. Of course, the same precautions are necessary for other systems trying to access your disk via Ethernet.

17.6 Remote Login, Remote Shells, Remote Copy

This section describes the commands `rlogin`, `rsh`, and `rcp` that are used for remote login, to execute a command on a remote host, and to make a copy remotely.

The `rlogin` command handles remote login on a remote host, for example:

`rlogin` *`rhost`*	Remote login on host *`rhost`* under same *`username`*
`rlogin` *`rhost`* `-l` *`username`*	Remote login on host *`rhost`* under new *`username`*

The `rsh` command allows executing commands on a remote host, for example:

`rsh` *`rhost command`*	Execute command on remote host *`rhost`* under the same user-ID
`rsh` *`rhost`* `-l` *`username command`*	Execute *`command`* on remote host *rhost* under a new *`username`*
`rsh` *`rhost`*	Same as `rlogin` *`rhost`*

The `rcp` command makes a remote copy. *Do not use it for local copying*. For example:

`rcp` *`source target`*	Remote copy, where *`source`* or *`target`* (or both) have form *`rhost:pathname`*
`rcp -r` *`source target`*	Recursive remote copy, where *`source`* and *`target`* must be a directories

Each command requires a correct setup of `/etc/hosts.equiv` or `~/.rhosts` (or both) on both systems involved.

17.7 Remote Printing and Plotting

Remote printing and plotting are as easy to set up as local output. The manual *VNMR and Solaris Software Installation* contains information. Just ensure that the printer and plotter definitions are consistent throughout the network. Even cross-platform printing and plotting should be fully functional, with few exceptions (we have seen occasional compatibility problems when remotely printing or plotting between SunOS 4.x and Solaris 2.x systems).

17.8 Advanced Networking: Connecting to the Internet

To connect to other networks, a number of additional steps must be undertaken. Besides telling the system in which fully qualified Internet domain it resides, we must establish mechanisms for the system to access IP addresses on other networks through a gateway computer (IP routing), and then we want to enable a mechanism for the system to convert fully qualified Internet Host names into numeric IP addresses (Domain Name Services, DNS).

Defining the Default Gateway (/etc/defaultrouter)

Your network administrator provides information on the Internet router to use. You should store the *numeric IP address* (such as 192.72.123.2) of the router (rather than the host name, even if the router is defined in `/etc/hosts`) in the file `/etc/defaultrouter`. When booting up next time, the system automatically accesses the router whenever you try communicating with an IP address that is not on the local network. You can also directly activate this IP routing using the command

```
route -f add default ‘cat /etc/defaultrouter‘ 1
```

Activating DNS (/etc/resolv.conf, /etc/nsswitch.conf)

With the above steps, you can now communicate with any system on Internet, provided you know its numeric IP address. Any host for which you know the IP address can be listed in `/etc/hosts`, and then it can be accessed with the host name rather than the numeric IP address. However, you may not always know the numeric IP address of the system with which you want to communicate. As numbers are difficult to memorize, people prefer addressing remote systems by their Internet host name, such as
`lal600.al.nmr.varian.com`

There is no global address book for translating such host names into IP addresses, but there are name servers running specialized software that gathers such information from the Internet. All you need to do for the local host is to tell it which name server to contact for resolving host names not listed in `/etc/hosts`. The primary step for achieving this is, to create a file `/etc/resolv.conf` of the format

```
nameserver   132.190.185.172
nameserver   192.76.144.66
domain ch.varian.com
```

The file contains one or more lines starting with `nameserver`, followed by the IP address of each name server (ask your system administrator about the name servers to use). These name servers are typically *not* on the local network. It is advisable to list the name server that is easiest to access on the first line, the secondary name server on the second line, and so on. Specifying multiple name servers is helpful in case one of these systems is temporarily down, or if one system has problems resolving an address. The last line (starting with `domain`) defines the local Internet domain name (same as in `/etc/defaultdomain`).

This alone doesn't yet activate DNS. The file `/etc/nsswitch.conf` defines where information for the various networking services can be retrieved from. On a stand-alone system (without NIS), this file (without comment lines) looks as follows:

```
passwd:          files
group:           files
hosts:           files
```

```
networks:       files
protocols:      files
rpc:            files
ethers:         files
netmasks:       files
bootparams:     files
publickey:      files
netgroup:       files
automount:      files
aliases:        files
services:       files
sendmailvars:   files
```

This means that for all these categories, the information is to be retrieved from the local files in `/etc`. All you need to do now is to change the two lines starting with `hosts:` and `networks:` to

```
hosts:          files dns
networks:       files dns
```

This tells the system to retrieve host and network information using DNS (as configured through `/etc/resolv.conf`). On systems using NIS, `/etc/nisswitch.conf` has a different format.

Of course, apart from systems on your own network, you can still only access systems that are accessible to the public. Some systems, such as most systems at Varian, are hidden behind a security door, typically firewall software, which allows connections from a local system to the outside world, but which blocks off incoming traffic (except perhaps for a few selected protocols, such as simple mail).

17.9 Mounting File Systems (mount, /etc/dfs/dfstab)

The mounting process allows building a file system or a part of a file tree into another file tree. Mounting requires two files—the file system to be mounted (it can be a disk partition or a partial file system such as `/usr/man`) and a mount directory onto which the file system is mounted. This process hides any previous contents of the mount directory, which only can be accessed by unmounting the file first. Therefore, mount directories normally are empty.

Local Mounting

What can mounting be used for? Mounting is already used on every UNIX machine. For example, `/dev/dsk/c0t3d0s5` is mounted onto `/usr`, and `/dev/dsk/c0t3d0s6` is mounted onto `/export/home`. `/usr` and `/export/home` are empty mount directories. These files are mounted as ufs type (UNIX file systems). This type of mounting is always used for a system's own disks (more exactly, disk slices or file systems). These disk partitions can also be mounted, explicitly, using the `mount` command:

```
mount -F ufs -o rw /dev/dsk/c0t3d0s6 /export/home
```

This would be the full command (it allows for a large number of options). The file type is specified as `ufs`, the partition is mounted with `rw` (read/write) permissions. For local disks, these are the default options, and a simpler command would do as well:

```
mount /dev/dsk/c0t3d0s6 /export/home
```

If this file system is listed in `/etc/vfstab`, which is usually the case (see below), even

```
mount /export/home
```

or

```
mount /dev/dsk/c0t3d0s6
```

is sufficient. Only block devices from `/dev` (not raw devices such as `/dev/rdsk/*`) can be mounted this way.

The `mount` command without arguments displays a list of the mounted file systems, for example:

```
> mount
/ on /dev/dsk/c0t3d0s0 read/write/setuid on Mon Oct 27 13:27:33 1997
/usr on /dev/dsk/c0t3d0s5 read/write/setuid on Mon Oct 27 13:27:33
1997
/proc on /proc read/write/setuid on Mon Oct 27 13:27:33 1997
/var on /dev/dsk/c0t3d0s3 read/write/setuid on Mon Oct 27 13:27:33
1997
/export/home on /dev/dsk/c0t3d0s6 setuid/read/write on Mon Oct 27
13:28:27 1997
/opt on /dev/dsk/c0t3d0s4 setuid/read/write on Mon Oct 27 13:28:27
1997
/disk on /dev/dsk/c0t2d0s2 setuid/read/write on Mon Oct 27 13:28:27
1997
/tmp on swap read/write on Mon Oct 27 13:28:27 1997
```

Unmounting a Local File System

The command `umount` enables the unmounting of mounted file systems, for example:

```
umount /dev/dsk/c0t3d0s6
```

The following simpler version works as well:

```
umount /export/home
```

Because all ufs-type mounted file systems are usually essential for stand-alone operation (and mounting is done automatically at bootup time), the `umount` command is rarely used for local file systems. File systems that are in use, such as the `/` and `/usr` file systems, can not be unmounted. A file system is in use as soon as a user or a shell in general has the working directory within it or as long as a program keeps a file open in that slice.

On desktop SPARCstations, floppies with UNIX file systems can be mounted the same way as hard disk partitions (note that the floppy must be mounted as a partition). Floppy disks with DOS file system require special options to be used with the `mount` command (see also “Using Floppy Disks,” page 156).

Remote (NFS) Mounting

With Sun computers (and most other UNIX workstations), the `mount` command has a very important extension: NFS-type mounting of remote file systems. NFS is Sun’s Network File System, a popular mechanism to extend a file system across a network. NFS mounting works very similar to ufs mounting:

```
mount -F nfs -o ro unity500:/export/home /home500
```

This mounts the `/export/home` file system of the host `unity500` in `ro` (read-only) mode onto the local directory `/home500`, which must exist before the `mount` command can be called. In this case, `unity500` is a file server for the local host, which is then called an NFS client machine. Note that NFS mounting only works if a file system is shared by the server system, see "NFS Mounting Options, Sharing a File System," page 236.

There are certain default values. For remote files, NFS-type mounting is default; thus,

```
mount unity500:/export/home /home500
```

mounts the same file with read-write permission (read-write is the default).

Unmounting a Remote File System

Unmounting works the same way as for ufs-mounted file systems:

```
umount unity500:/export/home
```

or simply

```
umount /home500
```

Accessing NFS files is more complex than accessing than local disks because NFS must provide for facilities that can cope with a broken Ethernet, an inactive file server, or a temporarily broken or overloaded Ethernet.

NFS Mounting Options, Sharing a File System

This leads to a number of new options for NFS type of mounting, including:

- Number of mount retries (`retry`)
- Whether or not to retry in foreground (`fg` and `bg`)
- NFS time-out (`timeo`, in tenths of a second)
- Number of retransmissions
- Whether to return an error if the server does not respond (`soft`) or to retry until the server does respond (`hard`)
- Buffer sizes

The defaults are `fg`, `retry=10000`, `timeo=7`, `retrans=3`, and `hard`.

Similar to the commands `rlogin`, `rsh`, and `rcp`, NFS can affect the security of a UNIX system in a network. Therefore, the operating system has a facility allowing specification of the file systems on the server that can be mounted by remote clients. This is achieved with the `/etc/dfs/dfstab` file. This is a simple text file that lists file systems that are shared with (and can therefore be mounted by) other systems. `/etc/dfs/dfstab` contains one line with a share command per shared file system, e.g.:

```
share -F nfs -o ro=unity:mercury /export/home
share -F nfs /data
```

This allows the `/export/home` and `/data` file systems to be mounted remotely. By default, file systems (such as `/data` in the above example) are shared with read-write permission (but the standard permission mechanism can still be used to secure individual files and directories). Also, by default, an shared file system can be mounted by every host on your network, not just in your local subdomain. If you want to share a file system with selected hosts only, you must use the following syntax:

```
share -F nfs -o ro /usr/share
share -F nfs -o ro=mercury /export/home
share -F nfs -o rw=i400:i500:mercury,ro=server /data
```

In this case, `/usr/share` is shared with any system, but is read-only; `/export/home` can be mounted read-only and by the named host only; and `/data` can be mounted read/write, but only by the listed hosts plus read-only by a system named `server`.

Note that options to the `share` command (`rw`, `ro`, etc.) must be separated by a comma, and in options with host name lists, the names must be separated by a colon.

Creating the file `/etc/dfs/dfstab` does not automatically share the file systems. If the file is newly created, the NFS and mount daemons (`nfsd` and `mountd`) are not even running yet. Rebooting the system is the safest and recommended way to achieve this. If you have already shared file systems before and were just adding another entry in `/etc/dfs/dfstab`, you can share the newly added file systems by entering

```
shareall
```

To see which file systems are currently shared on your system, enter `share` without any argument (this information is also found in `/etc/dfs/sharetab`). This does not indicate whether the file systems are actually mounted by any remote system. It is also possible to unshare selected file systems (if they are not currently mounted by any remote system) using the `unshare` command. For details, see `man unshare`.

If mounting doesn't work after entering `shareall` (and `share` indicates that the file system is really shared), most likely the NFS daemon or the `mountd` daemon (or both) are not running. Check the process table using `ps -ef` to see if `/usr/lib/nfs/nfsd` and `/usr/lib/nfs/mountd` are running. When you don't want to reboot the server system but still want to start sharing file systems, you can start the NFS and `mountd` daemons (as `root`, of course) by entering:

```
/usr/lib/nfs/nfsd -a 16
/usr/lib/nfs/mountd
```

Limitations

Note that `mount` makes files from many partitions, disks, and even computers look like a single tree structure. For most commands, this is really the case. For other commands, such as `mv` on directories, do not work between different computers or partitions and always recognize mount points.

With NFS, the user gets easy access to files on other machines, too easy in some ways. A user can move around in foreign directories, execute commands from remote disks, and can copy those onto the local disks. There is no check whatsoever whether compiled software is compatible with the user's hardware. Incompatibilities can lead to error messages and even serious crashes. Unless the server and the client both have the same architecture and operating system, such as two Sun workstations running Solaris 2.x, you need to be quite careful when dealing with compiled programs on remote disks.

NFS Mounting and `root` Access

The user `root` faces another problem. When working on mounted remote file systems, its user-ID is automatically reset to 60001, and its permissions correspond to those of nobody, which only gives access to files with permission for others. This prevents inadvertent file destruction by `root` on remote data systems. On the other hand, it also does not allow `root` to create files on remote disks. `Acqproc` is owned by `root` and, therefore, cannot acquire (write) into experiments located in NFS-mounted file systems. There is a remedy for this problem, however (giving up on some security features). In `/etc/dfs/dfstab` you can specify hosts from where `root` keeps root permission on exported and NFS-mounted file systems:

```
share -F nfs -o root=unity:mercury /data
```

gives `root` from `unity` and `mercury` root access on the remotely mounted `/data` file system. Therefore, users are allowed to have their experiments or home directories on that remote file system and do acquisitions into these experiments, see "Remote Home Directories," page 246. Note that multiple hosts are separated by a colon; the comma is the separator between multiple arguments.

17.10 Automatic Mounting (/etc/vfstab)

When UNIX is booted up into multiuser mode, the file systems `/dev/dsk/c0t3d0s0`, `/dev/dsk/c0t3d0s5`, and others are mounted automatically. On a standard system, this feature is used for all local disk partitions set up by the installation software when loading Solaris. The information as to what partitions need to be mounted is taken from the file `/etc/vfstab`, which on a stand-alone system with a single disk may consist of the following lines:

```
#device        device     mount    FS      fsck    mount     mount
#to mount      to fsck    point    type    pass    at boot   options
#
fd              -          /dev/fd  fd       -       no        -
/proc           -          /proc    proc     -       no        -
/dev/dsk/c0t3d0s1  -       -        swap     -       no        -
/dev/dsk/c0t3d0s0 /dev/rdsk/c0t3d0s0 /      ufs     1       no    -
/dev/dsk/c0t3d0s5 /dev/rdsk/c0t3d0s5 /usr   ufs     1       no    -
/dev/dsk/c0t3d0s3 /dev/rdsk/c0t3d0s3 /var   ufs     1       no    -
/dev/dsk/c0t3d0s6 /dev/rdsk/c0t3d0s6 /export/home ufs 2     yes   -
/dev/dsk/c0t3d0s4 /dev/rdsk/c0t3d0s4 /opt   ufs     2       yes   -
/dev/dsk/c0t6d0s0 -        /cdrom   hsfs     -       no        ro
swap               -       /tmp     tmpfs    -       yes       -
```

The meaning of the individual entries is reasonably obvious:

- Device to be mounted (a block device for disk slices)
- Device address used by the `fsck` command (a raw device for disk slices),
- Mount point (mount directory)
- Mount type (`ufs` for local disk slices)
- `fsck` pass number defining the sequence in which disk slices are checked (a hyphen `-` means no `fsck` on that file system)
- Flag (`yes` or `no`) that defines whether a slice or file system is to be mounted automatically at boot time
- Options (such as `ro` for read only, `rw` for read/write, or simply `-`) to select the default mount options for the given mount type).

The line starting with `fd` refers to a floppy drive that requires special mounting (see "Using Floppy Disks," page 156). Also the CD-ROM, `/dev/dsk/c0t6d0s0`, requires a special mounting mechanism (hsfs for the High-Sierra File System). File systems that are not mounted automatically can be mounted manually, using the `mount` command. Still it makes sense to have such entries, because then only the name of the mount directory or of the file system needs to be specified with the `mount` command.

The file `/etc/vfstab` can now be extended to also automatically include NFS mounted files, for example:

```
#device        device     mount    FS      fsck    mount     mount
#to mount      to fsck    point    type    pass    at boot   options
```

```
#
fd              -         /dev/fd   fd       -       no      -
/proc           -         /proc     proc     -       no      -
/dev/dsk/c0t3d0s1 -        -        swap     -       no      -
/dev/dsk/c0t3d0s0 /dev/rdsk/c0t3d0s0 /       ufs     1       no  -
/dev/dsk/c0t3d0s5 /dev/rdsk/c0t3d0s5 /usr    ufs     1       no  -
/dev/dsk/c0t3d0s3 /dev/rdsk/c0t3d0s3 /var    ufs     1       no  -
/dev/dsk/c0t3d0s6 /dev/rdsk/c0t3d0s6 /export/home ufs 2      yes -
/dev/dsk/c0t3d0s4 /dev/rdsk/c0t3d0s4 /opt    ufs     2       yes -
/dev/dsk/c0t6d0s0 -       /cdrom    hsfs      -       no      ro
swap             -        /tmp      tmpfs     -       yes
-server:/usr/share/man -  /usr/share/man
nfs   -       yes     ro,soft
unity:/data     -         /data      fs       -       yes     rw,bg
```

This also mounts the `/usr/man` directory from a remote host server. The UNIX manual is mounted `ro` and `soft`, and `/data` from the same host is mounted as well. The `fsck` pass number is not specified (`-`) because NFS mounted file systems are normally not checked with `fsck`. The `bg` option can be used to avoid that the bootup procedure gets stuck when trying to mount. If a file should temporarily not be mounted, unmount it manually and set the second-to-last item to `no` (no automatic mount at boot time).

To see what file systems are currently mounted (ufs and nfs), use `mount` without arguments. To mount all file systems specified in `/etc/vfstab`, you can use

```
mountall
```

The `mountall` command also tries mounting items such as the floppy disk and CR-ROM (if there are entries in `/etc/vfstab` for these devices). To mount only file systems of a specific type, use

```
mountall -F ufs
```

or

```
mountall -F nfs
```

which only mount the ufs or nfs type file systems from `/etc/vfstab`. This is no problem if files from `/etc/fstab` are mounted already. You don't need to unmount first because `mountall` makes the necessary additions.

The same options are valid for `umount`:

```
umountall
```

tries to unmount all mounted file systems. It does not unmount file systems currently in use, such as if a user currently has his or her working directory in a file system that is mounted. Only certain mount types can be addressed, which usually makes more sense:

```
umountall -F nfs
```

tries to unmount all NFS mounted files.

Finally, here are some hints and useful applications for `mount` and `umount`:

- Before calling `fsck` or `find /` (runs find on all accessible file systems), you don't necessarily want to cover NFS-mounted file systems. You could first unmount all NFS mounted files. Alternatively, you can use the `prune` option for NFS-type directories, for example:
  ```
  find / -name '*.fid' -print -fstype nfs -prune
  ```
  Note that the `-fstype nfs -prune` options should be specified last.
- If Ethernet or a file server is down while booting up, and you have NFS-type entries in `/etc/vfstab`, the system gets stuck while booting up. Press Ctrl-d to stop the

mount process, and the bootup will continue. Log in as `root` afterwards, and mount all NFS entries in `vfstab` with `mountall -F nfs`.

It is better to use the `soft` option for non-vital file systems like `/usr/share/man`, and the `bg` option for all other file systems that are to be mounted automatically. This allows the bootup process to proceed even in the case of networking problems.

17.11 Automounter

The automounter does more than automatic NFS mounting (standard NFS mounting is already automatic if set up via `/etc/vfstab`). The automounter is mounting on demand. It works with automount directories, such as `/home` or `/net`—special directories with an associated automounter map. As automounting only happens as needed (requested), it is more economical on system and network resources than standard NFS mounting. Also, automounting has been reported as being more robust and user-friendly in the case of temporary network problems.

How it Works: An Example

Your system has an automount directory `/net` (this is standard with Solaris 2.x), and we assume that you have a remote host `i500` on your network that is sharing the file systems `/data` and `/export/home`.

```
> cd /net
> ls
>
```

This indicates that right now there is no automatic mounting to `/net`. Now watch the following steps:

```
> cd /net/i500
> ls
data    export
> ls export
home
>
```

Note that the subdirectory `i500` was not there before we changed directory into it. The point is that mount points inside an automount directory as generated as you need them. The details of the automounting are fixed in an automounter map in `/etc`, see below. In this case, the definition is that shared directories from remote hosts can be accessed as `/net/`*`remotehost`*`/`*`shared_dir`*, which in the above example gives us the new paths `/net/i500/data` and `/net/i500/export/home`.

Again, this is a standard Solaris 2.x feature—all that is needed for this to work is that the remote system shares a file system first. *You can now see why we recommend sharing file systems with specified hosts only.*

If you list the contents of the remote directories, you see these files as if they were local. If your system is not granted access to certain shared file systems (see "NFS Mounting Options, Sharing a File System," page 236), you still see the subdirectories (`data` and `export/home` in the above example), but these directories would appear to be empty.

The automount subdirectories (mount points) are there for as long as we need them (e.g., as long as we work on files in this temporary file system hierarchy or as long as any user has his or her current working directory somewhere in the temporary directory structure).

When these directories and their subfiles are not used or accessed for more than five minutes (default), the automount daemon (`automountd`) automatically unmounts them.

Automount Maps (/etc/auto_master)

The central definition file for the automount daemon is `/etc/auto_master`. The relevant lines from this file are

```
+auto_master
/net     -hosts    -nosuid
/home    auto_home
/xfn     -xfn
```

The first line refers to an external NIS or NIS+ map and is not discussed further here (if an NIS or NIS+ map named `auto_master` exists, its contents are used in place of the above first line). The following lines stand for one automount directory each:

- `/net` automount directory – The `-hosts` option specifies that shared directories from remote NFS servers are to be mounted as described in the previous section. The `-nosuid` option prevents SUID operations (see "Special Permissions," page 122) and is an extra security feature.
- `/home` automount directory – A separate automount map, `/etc/auto_home`, is to be used.
- `/xfn` automount directory – A directory used in connection with the "federated naming system," It will not be discussed here.

The automounter map for `/home`, the file `/etc/auto_home`, in its default form contains only one single line for systems with NIS or NIS+:

```
+auto_home
```

This means that unless we have the corresponding NIS or NIS+ map set up, automounting in `/home` is not activated. If we want to use automounted home directories in `/home`, we need to expand `/etc/auto_home`:

```
+auto_home
user1    server:/export/home/user1
user2    server:/export/home/user2
```

It is advisable to have a local home directory, at least for `vnmr1` (see also "Remote Home Directories," page 246). In the above case (assuming `/export/home` on the server is shared), these directories are already accessible by using a path such as `/net/server/export/home/user1`. The path `/home` just provides for a simpler path than `/net`, at the expense of some extra setup work. You must have consistent user (user- and group-ID) definitions across your network for this to work properly.

17.12 Using FTP

NFS mounting is certainly the most convenient way of accessing remote files or making local files accessible to remote systems, especially if such connections are to be set up permanently. For a certain larger range of hosts, another option is to enable `rcp` for occasionally transferring files between two workstations. However, both these options require extra administrative overhead and a friendly administrator who sets up the networking parameters and permissions to make `rcp` possible.

In case you can't do `rcp`, you can still use FTP (file transport protocol) to transfer your data to a remote system. For transferring data to and from distant host, and in particular to download software and patches from a public FTP site, you have no other choice than to use FTP.

Using FTP Interactively

FTP can be used several ways: by using the `ftp` command directly or by using a Web browser or other graphical user interfaces, such as `ftptool` (public domain software). Web browsers and `ftptool` offer more comfort and security than using `ftp` directly. In particular, they automatically use the proper transfer modes. But as long as you observe some basic precautions, you can use the simple `ftp` command, even for multiple files.

Here are some tips on using the `ftp` command:

- By default, `ftp` operates in ASCII mode. This mode should only be used for ASCII files (normal text files, README files, pulse sequence source code, etc.). With binary files (compiled programs, compressed and tar files, binary data such as FIDs, etc.) you must use the binary mode; otherwise, the transferred data may be corrupted. Enter `bin` to switch to binary mode. Actually, ASCII files also work with the binary mode, so it is probably a good idea to use binary mode all the time.
- If you don't want to overwrite an existing file, you can specify with the `put` and `get` commands an optional target file name instead of the current working directory:
  ```
  get source_file target_file
  put source_file target_file
  ```
- For transferring multiple files from the same directory, use the `mget` and `mput` commands with multiple file names or wildcard characters (or both), for example,
  ```
  mget 5.3BacqSOLino103.* 5.3BpsgSOLino101.*
  ```
  With `mget` and `mput`, the target file name is always the same as the source file name.
- The `mget` and `mput` commands ask for a confirmation for every single file that they are about to transfer. This may be useful if a wildcard argument covers files that you don't want to transfer, but mostly it is unwanted and just slows down the transfer, requiring unnecessary user interaction. You can easily switch off (and on) this prompting with the `prompt` command. The interactive prompting can also be suppressed by calling `ftp` with the `-i` option. This option is particularly helpful when using `mget` or `mput` in `ftp` within shell scripts, see "Calling ftp Within Shell Scripts," page 244.
- Even without prompting, `mget` and `mput` still produce lots of output—reporting established data connections, transfer rates, file sizes, and so on. When transferring many small files, this is sometimes undesirable. With the `verbose` command you can toggle FTP between the silent and verbose modes.
- Most people don't know that `ftp` also has `mkdir`, `rmdir`, `delete`, `mdelete` and `rename` commands. All of these operate on the remote system.
- With `lcd` you can change the local working directory (the `cd` command within FTP is remote). For quickly inspecting local directory listings or for performing any other local UNIX command without quitting `ftp`, you can use a shell call via the exclamation mark as escape character:
  ```
  !ls -l
  ```

More information of `ftp` is found in UNIX manuals (use `man ftp`), or within `ftp` by using the `?` command for getting a list of available `ftp` commands.

Making FTP More Automatic

With FTP, you must first log in and specify a password. This can be simplified by creating a file `~/.netrc` that can contain a user name and, optionally, the corresponding password for an arbitrary number of FTP sites, e.g.:

```
machine www.nmr.varian.com login anonymous password
rolf.kyburz@ch.varian.com
machine vnmrnews.nmr.varian.com login usergroup password
vnmr4web
machine codonics1 login anonymous password 2
```

The format is the keyword `machine` followed by a host name (either a local host, a fully qualified Internet address, or a numeric IP address), the keyword `login` followed by a login name (for anonymous FTP, use `anonymous` as login name), and finally the keyword `password` followed by the appropriate password (for anonymous FTP sites use your e-mail address as the password). The `password` keyword and password is optional.

You can add as many entries as you like. If the file contains passwords, it must have permission 600; otherwise, `ftp` refuses to use it:

```
chmod 600 ~/.netrc
```

When calling `ftp` with a host name and password specified in `~/.netrc`, you can bypass the login procedure and directly start with the FTP transactions, for example:

```
> ftp -i www.nmr.varian.com
Connected to gabriel.
220 gabriel FTP server (SunOS 5.6) ready.
331 Guest login ok, send ident as password.
230 Guest login ok, access restrictions apply.
ftp> cd pub/patches
250 CWD command successful.
ftp> bin
200 Type set to I.
ftp> mget 5.3BacqSOLmer103*
200 PORT command successful.
150 Binary data connection for 5.3BacqSOLmer103.Readme
(132.190.187.200,40020)
(1472 bytes).
226 Binary Transfer complete.
local: 5.3BacqSOLmer103.Readme remote:
5.3BacqSOLmer103.Readme
1472 bytes received in 1 seconds (1.4 Kbytes/s)
200 PORT command successful.
150 Binary data connection for 5.3BacqSOLmer103.tar.Z
(132.190.187.200,40021)
(167569 bytes).
226 Binary Transfer complete.
ftp> !ls -l
total 2908
-rw-r--r--   1 vnmr1    nmr 1472 Jan  9 01:38
5.3BacqSOLmer103.Readme
-rw-r--r--   1 vnmr1    nmr 167569 Jan  9 01:38
5.3BacqSOLmer103.tar.Z
ftp> bye
```

```
221 Goodbye.
>
```

or in silent mode (using the `verbose` command), with less output:

```
> ftp -i www.nmr.varian.com
Connected to gabriel.
220 gabriel FTP server (SunOS 5.6) ready.
331 Guest login ok, send ident as password.
230 Guest login ok, access restrictions apply.
ftp> cd pub/patches
250 CWD command successful.
ftp> bin
200 Type set to I.
ftp> verbose
Verbose mode off.
ftp> mget 5.3BacqSOLmer103* 5.3BpsgSOLmer101*
5.3BgenSOLall101*
ftp> bye
>
```

Calling ftp Within Shell Scripts

With the presence of a `~/.netrc` file, it even becomes possible to call `ftp` within a shell script and without user interaction. An example where this could be used is with printing on a Codonics dye sublimation printer. These devices are driven by a SPARC engine, and plots are typically submitted through FTP. The user name is arbitrary, but the password is a number that specifies what print options are to be selected (e.g., 2 is a typical value for printouts in landscape format).

The `~/.netrc` file shown above includes an example line for such a printer, named `codonics1`. The following shell script `imagepr` permits submitting a plot to that printer by specifying the name of the file to be printed (typically a Sun raster or a PostScript file):

```
#!/bin/sh
ftp codonics1 << +
bin
put $1
bye
+
```

This script (make sure it is executable) can now be called with a file name, for example:

```
imagepr image.rs
```

We could also suppress the `ftp` output to make this script silent:

```
#!/bin/sh
ftp codonics1 > /dev/null << +
bin
put $1
bye
+
```

17.13 NIS and NIS+

NIS, the Network Information Service (initially called the "yellow pages") is a database for networks that allows for an easy maintenance of system administration files throughout a network. NIS covers files such as `/etc/passwd`, `/etc/shadow`, `/etc/group`, `/etc/networks`, `/etc/hosts`, and many others. The advantage is that all those files are centralized and need only to be maintained on a single host, whereas the others fetch this information via network.

With Solaris 2.x, NIS is being replaced by the newer **NIS+,** which offers enhanced security by using encryption and other features. NIS+ retains some back-compatibility with NIS.

Within an NIS/NIS+ domain, which or may not may not include all hosts within a network, there are three classes of hosts:

- NIS master server
- NIS slave servers
- NIS clients

The slave servers improve network safety by maintaining a duplicate of the master data base, the case the master is not operative.

Unfortunately, setting up an NIS+ service is a rather complex operation and covering that is far beyond the scope of this manual. Refer to the Solaris documentation *Name Services Administration Guide* and *Name Services Configuration Guide* for more information.

17.14 Heterogeneous Networks

Ethernet can be used to talk to computers other than Suns, of course. In fact, various configurations allow communication via Ethernet:

- The other system is a UNIX system and has NFS. Such an implementation should be straightforward.
- The other system is a UNIX system and does *not* have NFS, but TCP/IP is present. The implementation should still be simple. Commands like `rlogin`, `rsh`, and `rcp` work, but not NFS mounting of file systems.
- The other system is a non-UNIX system, such as a VAX under VMS. Either the other system requires TCP/IP or you'll need to have special software on the Sun to talk to the non-UNIX system. For example, TCP/IP is available for VAX computers but is expensive. The less expensive DECNET for the Sun achieves similar results but you have to review appropriate manuals to operate it.
- Other computers without standard operating systems—such as Gemini, VXR, and XL spectrometers—require special software (such as the Varian limNET software in the case of these systems) for both the remote computer and for the Sun to allow for some lower-level but still fast file transfers.
- TCP/IP, `ftp`, and `telnet` are also available for Apple Macintosh computers, IBM PCs, and PC compatibles. Therefore, a high-level protocol can be used, even with these systems.

17.15 Remote Home Directories

In a network, it is not necessary to have a home directory on each system for each user. Every user can basically have a single home directory. How can this be achieved?

Using the Automounter

Setting up remote home directories is relatively easy with the automounter, see "Automounter," page 240. You set up an automount map `/etc/auto_home` according to the following scheme:

```
vnmr1   server_name:/export/home/vnmr1
vnmr2   server_name:/export/home/vnmr2
```

provided that on the server system the file system `/export/home` is shared, see "NFS Mounting Options, Sharing a File System," page 236. You can then define these users with the same user- and group-ID as on the server and with a home directory in `/home` (rather than `/export/home`), using the Solaris AdminTool.

In the case of network problems, however, users with automounted home directories cannot even log in. If everything works as it should, a user's home directory is mounted automatically when the user logs in, and it is automatically unmounted five minutes after a user logs out.

If you prefer a solution with static mounting and with a possible fall-back position in the case of network problems, see the next section.

Using Standard NFS Mounting

For standard NFS mounting, on each system where that user should have access, the password and group files (and eventually other files that regulate the access to certain files or systems) must be updated (this is easy with NFS). Then, an empty home directory is generated on all systems (`/home/`*`username`*). This home directory might contain some default files (dot files, eventually also a small `vnmrsys` for VNMR) that would allow the user to log in and work, even if the network cannot be reached. The path to that home directory may be the same on all systems. This would allow fetching all information in the password file via NIS (see "NIS and NIS+," page 245).

Next, on each remote system, the full home directory must be mounted onto that directory. The mount should be read-write (type NFS, of course) and, therefore, hard mounted. The corresponding entry in `vfstab` would be, for example:

```
unity:/export/home/evan - /export/home/evan nfs - yes rw,bg
```

Once the home directory is mounted with the command `mountall -F nfs` or by rebooting, the contents of the default directory on the local machine are hidden so they can only be accessed by unmounting the NFS-mounted home directory exported from the remote host. The home directory on the remote host can be accessed like a local home directory. `/etc/dfs/dfstab` on the file server must allow mounting the home directory remotely.

Alternatively, if the whole remote disk or at least the file containing the home directory is NFS mounted (it would have to be hard mounted, with read-write access again), a simple symbolic link can be created instead of a default home directory:

```
ln -s /sun/home/evan /export/home/evan
```

This would be less safe solution, in that it would not provide for a home directory if the network is down.

Note that if a spectrometer host is involved, `Acqproc` must have write access to the home directory. There can be problems if the disk with the active experiment is not local to the spectrometer. This can be fixed by exporting the file system with `root` access for remote systems, see "Mounting File Systems (mount, /etc/dfs/dfstab)," page 234. Without that, the acquisition process, which is owned by `root` and has only "other" permission on remote systems (its user-ID is reset to 60001 for "nobody" on the remote host) cannot access the experiment files unless permissions are opened up dramatically (which is undesirable).

17.16 Gateway Machines

A *gateway machine* is a CPU that can talk to two different networks, usually on two Ethernet cables. It links two different networks and allows all users of the linked networks to talk to each other. This can be repeated on the added-on network and allows creating large scale networks.

What are the requirements for the gateway machine?

- It requires two Ethernet controllers. On Sun workstations, this means an additional Ethernet controller board. For installation of the extra board, see Sun manuals. Obviously, systems without a free expansion slot cannot be made a gateway system.
- It has two identities—one network number and one host name per network

The file `/etc/hosts` on a gateway contains two lines for the local CPU, defining two unique host names for both networks:

```
192.9.200.1         sun loghost       #(ie0)
192.9.201.33        vnmr-net          #(ie1)
```

The same file also contains the network addresses for all additional systems the gateway should talk to.

To activate the Ethernet hardware (through `rc.boot` at bootup time), you need an equivalent to `/etc/hostname.le0`. The file is named `/etc/hostname.le1` for standard Ethernet interfaces and contains the host name for the other network (`vnmr-net` is used in our example).

After these changes, the gateway must be rebooted using `boot -r`. The `/etc/hosts` file also must be updated on all other systems to include the newly accessible network addresses, unless the gateway is also the NIS master server.

Spectrometer Hosts Should Not Act as Gateways!

From a hardware configuration point-of-view, host computers of networked UNITY*INOVA*, *MERCURY*-Series, and *GEMINI 2000* spectrometers are also gateway computers, because they have two Ethernet interfaces. We do not activate any gateway functionality, however, because the acquisition network is not intended for communication with other networks. Creating a file `/etc/defaultrouter` ensures that no Internet routing or name daemon (`in.routed` or `in.named`) is started on a spectrometer host. If any of these daemons is running by mistake, the spectrometer host can be severely slowed down. An even safer way to suppress unwanted routing activities is to create a file `/etc/notrouter`, by entering:

```
su
touch /etc/notrouter
```

17.17 Diskless Systems

Solaris permits setting up servers that store all the disk-based information for other Sun computers on local disks. These other systems then become diskless clients, because they don't have any local disks. The server knows the Ethernet hardware address for these clients. Upon booting, a client broadcasts its Ethernet address, and the server responds with its own and the client's IP addresses. The client then downloads the booting software, followed by the kernel (from its dedicated root partition on the server), and finally it starts UNIX, using shared partitions on the server.

Such a setup has some advantages:

- All disk-based information is stored, administered and backed up at a central location. Once it is set up, the system administration is centralized and easier.
- All user software needs to be installed only once.
- Because most of UNIX is stored in shared partitions and all system share the home directory and data partitions, disk space and disk hardware is saved. But because disk space is no longer excessively expensive, this argument loses much of its validity.

But there are also disadvantages:

- The initial setup is considerably more complex.
- There are performance losses due to network speed limitations.
- With fast networks, the disk might become a performance bottleneck.
- Such a setup is more vulnerable. If either the server or the network is down, the clients are down as well.

Because such configurations are now used even more rarely than a few years ago (when disks were much more expensive relative to the costs of a workstation), this topic is not discussed further in this manual.

Chapter 18. X Window System Operation

Sections in this chapter:

- 18.1 “X Window System Environment,” this page
- 18.2 “X Window System Operation,” page 250
- 18.3 “Using a Workstation as an X Server,” page 252
- 18.4 “Using a Personal Computer as an X Server,” page 254
- 18.5 “Customizing the X Environment,” page 258

X Window system VNMR software from Varian exists in several variations, each operates with a different graphical output:

- VnmrX on Sun computers with the Open Look window system.
- VnmrI on IBM RS/6000 computers with the Motif window system.
- VnmrSGI on Silicon Graphics computers with the Motif window system.

All VNMR products can send their output to certain graphics terminals either directly attached to one of the RS-232 ports of the computer or in some way accessible to the computer over a network (serial or Ethernet). The terminals supported for graphics include the Tektronix models 4105, 4107, 4205, and 4207, and the GraphOn models 235, 240, and 250. Terminal emulation programs that run on Macintosh or IBM PC computers can also be used, although users should be aware that terminal emulation is an imperfect science, and a program that emulates 99 out of 100 commands will not be useful with VNMR if VNMR relies on the remaining command.

Other terminals or terminal emulators can also be used to operate VNMR in a “dumb” mode (i.e., in a mode that does not support graphics). While this is obviously less than desirable, in some situations it may be perfectly reasonable—for testing macros, preparing parameter sets, and so on.

18.1 X Window System Environment

VnmrX, VnmrI, and VnmrSGI are X Window system products. In the X framework, one computer, referred to in this chapter as the *X host*, is the computer that runs the software (this computer is also called the *X client*). Either the same computer or another computer acts as the *X server*, which is the computer on whose screen the graphics output of the client appears and whose mouse and keyboard are used for input. An X host can support multiple X servers, which means that a large number of users can be operating the software from multiple remote sites (the same is true using Tektronix or GraphOn terminal graphics).

X servers can be a variety of things:

- One possibility is that the same computer that is the X host is also the X server. In this case, graphics are displayed on the screen of the same computer in which computations occur.

- A second possibility is that a second computer is used for this task. The second computer has to be equipped with X server software that allows it to function in this manner. If the second computer is a Macintosh or PC computer, X server software must be purchased to allow that computer to receive X output.
- The final possibility for an X server is a dedicated *X terminal*, a special-purpose terminal programmed to receive and interpret commands in the X language.

All operation of VnmrI and VnmrSGI from terminals and X servers is fundamentally data station operation (i.e., data processing only). Limited control over data acquisition is possible, assuming that the computer running the software (not the computer displaying the graphics) is connected to the spectrometer. The remainder of this chapter describes only *differences* in operation of these products from VnmrX.

Note: Varian neither sells nor endorses X Window system products for the PC or Macintosh. Please contact the software companies selling these products or their dealers for information, purchase, and support.

18.2 X Window System Operation

The VnmrI and VnmrSGI versions differ from VnmrX in the following ways:

- *VnmrI*, *VnmrSGI* – The `acqi` command is not active.
- *VnmrSGI* – A C compiler is optional on Silicon Graphics systems. If it is not present, the `seqgen` and `wtgen` commands will not work.

Starting and Exiting VnmrI

The following sections describe how to start and exit the VnmrI software version.

To Start VnmrI

Assuming you are using an account that has been configured properly (see the manual *VNMR and Solaris Software Installation* for a description of this process), the VnmrI program can be run *one* of the following ways:

- When you first log in to the data station, your `.login` file is executed. The system prompts:
  ```
  Auto Window start (^C aborts)
  ```
  A Motif Window Manager window (filled with the Varian icons) opens, and VnmrI starts automatically. At the same time, a second virtual terminal showing a console window (a “vanilla” window in which a window manager is not operating; usually showing green letters on a black background) is accessible; you can switch between the Motif window with VnmrI and the console window by pressing simultaneously the Alt and Ctrl/Act keys.

  If you press Ctrl-C, the login process halts with you in the console window.
- From a console window, enter `vnmr` or `vnmri` (but not `vnmr&`) to start the Motif Window Manager and VnmrI.
- From a window within a Motif environment (the window labeled VNMR or any other), enter either `vnmr` or `vnmri` to start VnmrI and then leave that window active.

If you find the mouse sensitivity too great when running VnmrI, you may wish to edit your `.login` file and change `xinit –bs` to `xinit –bs –a 2` to achieve a more controlled mouse movement.

Running Multiple Copies of VnmrI

More than one copy of VnmrI can be started in a single virtual terminal using the third method described above (under "To Start VnmrI") although, in general, this will be confusing. A preferable method is to open a second copy of VnmrI in a different virtual terminal as follows:

1. Press Alt and Ctrl/Act simultaneously to switch to the second virtual terminal.
2. Enter `vnmr` to start the Motif Window Manager and VnmrI. Alternatively, enter `open csh` to open a third virtual terminal.
3. From the console window in either of the two virtual terminals not running Motif, enter `vnmr`.

Again, switch between the various virtual terminals using the Alt and Ctrl/Act keys (or the Shift and Ctrl/Act keys to switch in the opposite direction).

Unless you log in as a different user in the second virtual terminal, the limitation you face is that the two or more copies of VnmrI now running cannot both be in the same experiment.

When programs are running in different virtual terminals, processing occurs in all virtual terminals—the one which is visible, and any others that are present. Graphical operations however (e.g., `dpcon`) only occur in the visible virtual terminal. Thus, if you start a large transform in one experiment in one virtual terminal, and then switch to another virtual terminal to work on data in another experiment, when you return to the first virtual terminal the processing will probably be finished, but the display of the data will not begin until the moment you reactivate that terminal.

To Exit VnmrI

- To exit from VnmrI, enter `exit`.
- To exit from the Motif Window Manager, press the Ctrl, Alt, and Backspace keys simultaneously (not Ctrl-Alt-Del as in DOS).
- To close a second virtual terminal that is in a console state (Motif has already been exited), press Ctrl-D.
- When you return to a console window, you may find the terminal characteristics are altered so that new lines do not return to the left edge of the display. If this occurs, enter `stty sane` to fix the problem.

Starting and Exiting VnmrSGI

The following sections describe how to start and exit the VnmrSGI software version.

To Start VnmrSGI

Assuming you are using an account that has been configured properly (see the manual *VNMR and Solaris Installation* for a description of this process), the VnmrSGI program can be run in one of the following ways:

- When you first log in to the data station, your `.login` file is executed, a 4Dwm Window Manager window (filled with Varian icons) opens, and VnmrSGI automatically starts.
- From a window within a 4Dwm environment (either the one labeled VNMR or any other), enter either `vnmr` or `vnmrsgi` to start VnmrSGI and leave that window active.

Moving Windows within VnmrSGI

If you use the default software configuration, VnmrSGI appears with windows lacking the usual Motif title bar that simplifies moving, closing, and resizing windows. The software can be configured to display the title bar, but if you choose not to do so, it is still possible to move, close, and resize the VnmrSGI windows.

To perform any of these operations, move the cursor to anywhere on the thin border that surrounds the window—it will change from an arrow to another shape.

- To resize the window, hold down the left mouse button and drag the window to the desired size. This works for the graphics window; however, you must redisplay anything that was on the screen. Do not resize the text entry and display windows.
- To move the window, hold down the center mouse button and drag the window to where you want it.
- To perform other operations, hold down the right mouse button and select the desired action from the menu. You should not select Close or Quit because these will terminate the program in an unexpected and undesired way.

To Exit VnmrSGI

- To exit from VnmrSGI, enter `exit`. Do not use Window Manager commands to close or quit.
- To exit from the 4Dwm Window Manager and log out, select Log Out from the drop down menu.

18.3 Using a Workstation as an X Server

Several similar operations exist when using any workstation (Sun, IBM RS/6000, SGI, and probably other, untested workstations as well) as an X server for any of the versions of VNMR— VnmrX, VnmrI, or VnmrSGI.

Setting Up a System

Of the several ways to configure your system, the following method is the simplest:

1. Enter `xhost +` on the workstation you will be using as an X server, which allows remote hosts to use your computer as an X server, or just enter `xhost` *`hostname`* to allow a single remote host to be used.
2. From an X environment (Motif, Open Windows, or whatever is available on your workstation), use the `rlogin` command to remotely log on to that computer.
3. You are asked about the type of terminal you are logging in from:
   ```
   Choose x)window t)ek d)umb g)raphon [x/t/d/g(default]:
   ```
   Enter `x`.
4. You are also asked to enter the name of your display server:
   ```
   input display server name:
   ```
   Enter the name of your computer.
5. Enter `vnmr` to start VNMR (the X client) on the remote host and display information on your computer (the X server).

The `xhost` command did not execute correctly if you get messages such as the following on the local window:

```
Xlib: connection to "varian:0.0" refused by server.
Xlib: client is not authorized to connect to server.
```

Some general difficulties you may encounter have to do with the interpretation of different keys. When you are working on a remote computer in a general window (shell), the Delete, Backspace, and Ctrl-H keys may all function differently than you expect. If you make a mistake when typing, the one key combination that seems to work universally is Ctrl-U, which erases the entire line you have entered. Within VNMR itself, however, all keys should function normally.

An alternative method of running VNMR remotely is to execute a remote command string on the workstation in order to start VNMR, without ever logging in directly to that workstation. To do this, the file `.xlogin` is provided with your VNMR software to function as an alternative to the standard `.login` file. The `.xlogin` file sets all of the environmental variables required for proper functioning of VNMR. Therefore, a remote command string can be set up to execute both this file (`source .xlogin`) and the `vn` command to start VNMR. This command string supplies the display variable as an argument:

```
rsh remotehost "source .xlogin; vn –display xserver:0.0"
```

substituting computer names as appropriate, for example,

```
rsh u500 "source .xlogin; vn –display ipx:0.0".
```

Using the `rsh` method of running VNMR remotely requires proper setup of UNIX files, including `/etc/hosts.equiv`. Enter `man rsh` from UNIX for more details.

Some specific difficulties, generally having to do with font availability, are described in subsequent sections of this chapter.

Using the RS/6000 as an X Server

IBM RS/6000 computers running AIX have a virtual terminal mode in which more than one screen (up to 16) can be made available and accessed successively by simultaneously pressing the Alt and Ctrl/Act keys. In this mode, it is essential to know the display number of the particular virtual terminal from which you are going to remotely activate VNMR running on another computer.

Before using `rlogin` to log in to the remote computer, but after starting the Motif Window Manager on the RS/6000, open a window and enter `env` to display a list of environmental variables. Look for the entry that reads `DISPLAY unix:n`, where `n` is a number from 1 to 16. Now enter `rlogin` as described above, and respond to the questions, but before starting VNMR, enter

`setenv DISPLAY` *`hostname`*`:`*`n`*`.0`

substituting the host name of your RS/6000 for *`hostname`* and setting the number *`n`* as appropriate (e.g., `setenv DISPLAY myibm:2.0`). This step is not necessary if `n=1`. Now start VNMR by entering `vnmr`.

Using the RS/6000 as an X Server with a Sun as Host

The fonts required by VnmrX (the software running on your Sun) are not present on the IBM RS/6000. Included in the `/vnmr` directory is a VnmrX subdirectory named `xvfonts`, which includes instructions and UNIX tools to help get the necessary Sun fonts into the IBM RS/6000.

Briefly, Sun fonts are converted into a portable format with the VNMR command `getxvfonts`. These fonts are then transferred to the IBM system and added to the IBM font list with the VNMR command `addxvfonts`. Detailed instructions are included in the `README` file in the `xvfonts` subdirectory.

Using the Sun ELC as an X Server to VnmrI

If you are using a monochrome Sun ELC computer as an X server for VnmrI, you have to edit the `.Xdefaults` file on your Sun, changing `*VNMR* foreground` to `Black` and `*VNMR*background` to `White`.

Using a Sun as an X Server to VnmrSGI

If you are using a Sun computer running OpenWindows as an X server to VnmrSGI and you do not want title bars on the VNMR windows, you need to add the following line to the dot file `.Xdefaults` on your Sun:

```
olwm.MinimalDecor: master VNMR
```

You should also set `*VNMR*clientDecoration` in the .`Xdefaults` dot file on your Silicon Graphics computer to `-title`.

18.4 Using a Personal Computer as an X Server

Personal computers running a version of UNIX generally function similarly to that described above for workstations. The vast majority of personal computers are not running UNIX, however. To function as an X server for VNMR running on a remote workstation, a personal computer needs X server software. Such programs include MacX and eXodus for the Macintosh, and eXceed and PC-Xview for the IBM PC and compatible computers.

Within most X server software packages, two environments exist—rooted and rootless. The *rooted environment* means that all of the windows opened by VNMR (or other programs that you run) are contained within a single, larger window. By contrast, in a *rootless environment*, all of the X windows are separate windows. Both environments are acceptable, although you will probably find the rooted environment more convenient.

Starting Up VNMR Using a Personal Computer

Once you are running PC X server software on your personal computer, four VNMR start-up methods are available:

- Execute a blind command string on the workstation to start VNMR. To do this, the dot file `.xlogin` is provided with your VNMR software as an alternative to the standard `.login` file. The `.xlogin` file sets all of the environmental variables required for proper functioning of VNMR. Thus, a remote command string can be set up to execute this file (`source .xlogin`) and then to execute the `vn` command to start VNMR, supplying the display variable as an argument (e.g., `source .xlogin; vn -display pcname:0.0`).
- Remotely log in to the host computer using the telnet protocol if available, and then execute a command string similar to that described above for a blind command string.
- Start a terminal window (`xterm`) using the blind command mechanism, and then, from that window, start VNMR.
- Start an `xterm` window; then start a remote window manager such as Open Look (`olwm`) on the Sun, or Motif (`mwm`) on the RS/6000. Next, start VNMR either with typed commands or with choices from a suitably configured drop-down menu.

Most X server software offers a choice between two command execution protocols—RSH and REXEC. These protocols differ only in password requirements—RSH does not require a password while REXEC does.

`xterm` operation requires the least overhead and setup. If VNMR is started from an `xterm` window, VNMR is displayed on the server with fixed windows. With `olwm` or `mwm` running as the windows manager on the client, the server screen looks like the screen on the Sun when `olwm` is running on the Sun, or it looks like the screen on the IBM RS/6000 when `mwm` is running, or like the screen on an SGI when a 4Dwm Window Manager is running.

Starting VNMR in either the `olwm` or `mwm` environment provides sizable and moveable windows. In the `olwm` environment, the VNMR windows can be collapsed to individual icons by pressing F7 on the server keyboard.

Setting Up Telnet

The telnet protocol takes several steps, but it is useful during the initial set up of the server and client, and it provides confirmation of network communications between the systems.

If you are starting the client using a telnet session, the X server is started in the passive mode with the X server software running on the server, awaiting instructions from the client. To communicate with the client and instruct it to display back onto the server, you open the communications to the client through a telnet session.

1. After you log in, the following is displayed on the system that will become the server:
   ```
   Choose x)window t)ek d)umb g)raphon [x/t/d/g(default)]:
   ```
2. Choose `x` to start the X environment. You are then prompted to enter the name of your X server:
   ```
   input display server name:
   ```
3. To start a client on a Sun and display a simple `xterm` window on the server, the PC in this case, enter the following command line on the client:
   ```
   /usr/openwin/bin/xterm -display servername:0
   ```
   where *`servername`* is the name of your X server.

   To start a client on an IBM RS/6000 and display a simple `xterm` window on the server, the PC in this case, execute the following command line on the client:
   ```
   /usr/lpp/X11/bin/aixterm -display servername:0
   ```

To start a client on an SGI and display a simple `xterm` window on the server, the PC in this case, execute the following command line on the client:

```
/usr/bin/X11/xterm -display servername:0
```

At this point an `xterm` window displays on the server. You can also edit the `.login` file to include the above instructions and create a new choice, `p`, for a PC X terminal login. The following is an example of such a modification.

1. Copy the `.login` file to a backup file (e.g., `bkup.login`).
2. Edit the `.login` file by first finding the following line:
   ```
   echo -n 'Choose x)window t)ek d)umb g)raphon[x/t/d/g(default)]:
   ```
3. Edit the line to include the following (keeping the entry on a single line):
   ```
   echo -n 'Choose x)window p)cXterm t)ek d)umb g)raphon  \
   [x/p/t/d/g(default)]:
   ```
4. Locate the line:
   ```
   else if ($a = t) then
   ```

Then insert the following lines above it (replace `servername` as appropriate):

```
else if ($a = p) then
   set term=sun; setenv graphics sun
   setenv DISPLAY servername:0
   /usr/openwin/bin/xterm&
```

The choice to log in as a PC and display the `xterm` window back on the PC is now available and is identified in the line `setenv DISPLAY servername:0`.

The only X server that can use this login is the machine specified in *`servername`*. This feature is useful if limited access is desired. If many X servers are logging on to an individual account, using `.xlogin` may be more desirable than the limited access.

Starting the Client

If you have set up passwords and permissions, you can start the client and display the `xterm` window on the server using either the REXEC (you are prompted for a password) or the RSH (no password prompt—a trusted user login) protocol. In either case, the following instructions are required. These instructions are transmitted to the client by the server when the RSH or REXEC protocol is initiated.

To start a client on a Sun and display a simple `xterm` window on the server, the PC in this case, the server instructs the user to enter the following command line on the client:

```
source .xlogin; /usr/openwin/bin/xterm -display servername:0
```

To start a client on an IBM RS/6000 and display an `xterm` window on the server, the PC in this case, enter the following command line to be executed on the client:

```
source .xlogin; /usr/bin/X11/xterm -display servername:0
```

If you are not using the `source .xlogin` script, then add `–ls` after `xterm` (one space between `xterm` and `-ls`), and then a space and the rest of the command. The option `–ls` following `xterm` instructs the client to start a login shell and display it in an `xterm` window on the server. The `xterm` window can be further defined to include scroll bars, change the foreground and background colors, and so on.

The `login` command can be expanded to start VNMR immediately by adding `-e vn&` to the end of either of the above command lines. The option `–e` followed by the command `vn&` must be the last statement in the line. The effect is to log onto the client and be placed in the VNMR environment without any further effort and to start VNMR in the background. The `xterm` window, from which VNMR is started, is available to start other processes. For simple `xterm` operations, this is not as useful as it appears. Since the windows cannot be moved or sized, the `xterm` window is covered by the VNMR window. If you choose to start a remote window manager, adding the & to the end of the instruction to begin VNMR is important.

Making the Display Resemble a Window Manager

If desired, the server display can resemble the `olwm` or `mwm` display of the client system. This is accomplished by adding the following instructions to the command line for the system on which you are starting a client. (This procedure may not work on some systems.)

- If a Sun is the client and `olwm` is the environment desired, add the following after `servername:0`:
  ```
  -e /usr/openwin/bin/olwm -display servername:0
  ```
- If an IBM RS/6000 is the client and `mwm` is the environment desired, add the following after `servername:0`:
  ```
  -e /usr/bin/X11/mwm -display servername:0
  ```

With a window manager running, VnmrX can be started by opening a window and entering `vn&` or, in the case of `olwm`, by editing the file `.openwin-menu` and adding the line `"VnmrX..." exec vn.`

The drop-down `olwm` menu will now have an option for VnmrX. Clicking on the VnmrX menu option starts VnmrX.

Controlling xterm Window Appearance

The `xterm` window can take on a number attributes depending upon your needs. The following examples of scripts can be made into executable files. As executable files, they can be run during the RSH or REXEC start up of the X server. If you use a telnet login, you do not need to change the scripts to executable files because `source .`*`filename`* always works (unlike RSH and REXEC).

In these examples, note that when a line needs to be broken in a script, a backslash (\) is placed at the end of each part of the line.

Example 1. An xterm window started with a login shell

```
goxterm -ls
#   Xterminal session on the PC
      #
      /usr/openwin/bin/xterm -sb -bg black -fg cyan \
      -bd blue -cr yellow -ms red -b 3 -bw 3 -ls \
      -display servername &
```

In this script, the `xterm` window has scroll bars (shown by `-sb`), black background (`-bg black`), cyan foreground (`-fg cyan`), blue border (`-bd blue`), yellow cursor (`-cr yellow)`, red pointer (`-ms red`), internal border width of 3 pixels (`-b 3`) and window border width of 3 pixels (`-bw 3`) and begin a login shell (`-ls`). The `xterm` window is displayed on the X server in the background (&), `<servername>` `(-display <servername>:0&)`

Example 2. VnmrX started in an xterm window upon login

```
vnxterm
#   Xterminal session on the PC and VnmrX startup
      #
      /usr/openwin/bin/xterm -sb -bg black -fg cyan \
      -bd blue -cr yellow -ms red -b 3 -bw 3 \
      -display <servername>:0 -e vn -fn 8x13 &
```

Along with the instructions defining the `xterm` window, but excluding the login shell instruction (`-ls`), an additional command to start VnmrX (`vn`) in the background (&) with an 8x13 font size (`-e vn -fn 8x13&`) is added as the last command.

Solving Networking Problems

Some networks use a host address file to list all the machines and their addresses on the network. Other networks use a server to store all the machine names and addresses (e.g., a domain name server). Problems, such as “unknown host,” which causes the server to terminate and not make the connection, can occur with the X server software.

If nominal login and network functions are working, such as dumb terminals `login`, `rcp`, and so on, and you are using a domain name server, the problem might be the inability of the X server to access the host names in the domain name server (assuming no errors in the configuration files for the X server host and client).

A solution to the unknown host problem is to create a host address file in the network directory that has the name and network address of the machine you are logging onto and as well and the name and address of the X server. For each machine you intend to start as an X client, you need that machine's name and network address added to your host file.

18.5 Customizing the X Environment

Some VNMR programs have X resource setup files in the `/vnmr/app-defaults` directory that define the parameters of the program windows. These parameters include colors of the objects in the windows, font size and style of the lettering, and the size and location of the window. The files use ASCII text to set the values of the parameters. Editing the files can be done by any UNIX text editor such as `vi` or `textedit`.

For example, Figure 24 shows the default X resource file for the `qtune` command. The full path of the file is `/vnmr/app-defaults/Qtune`. An exclamation mark (`!`) at the start of a line in the file indicates that the line is a comment. Because a comment line can be placed anywhere, a common use of the exclamation mark is to identify an alternate choice to the previous line, as in lines 3 and 4 of `Qtune`:

```
*background:  grey90
!*background  lightsteelblue1.
```

By simply changing the exclamation mark from line 4 to line 3, the background color is changed from `grey90` to `lightsteelblue1`.

System defaults file for the parameters are provided in `/vnmr/app-defaults` during installation. Users can override the system defaults for group or individual use by creating a similar directory in `~vnmr1` or `~user`.

```
!@(#)Qtune 1.2 10/25/9X
*QtuneAppDefaultsVersion: 1
*background: grey90
!*background: lightsteelblue1
! Note: inputFocusColor is different than background,
! but visually identical
*inputFocusColor: #e6e6e6
!*inputFocusColor: #cbe1ff
!
*TextField*inputFocusColor: black
*TextField*blinkRate: 0
*MessageWindow.background: white
*MessageWindow.linesVisible: 5
*MessageWindow.width: 250
*HelpWindow.background: white
*HelpWindow.linesVisible: 30
*HelpWindow.width: 600
*HelpWindow.font: 7x13
*AxisColor: green
*GridColor: forestgreen
*DataColor: cyan
*CursorColor: yellow
*Marker1Color: red
*Marker2Color: hotpink
*Marker3Color: magenta
*TuneDisplay.background: black
!*TuneDisplay.background: blue4
*DisplayGeometry: 800x500+4+25
*DisplayFont: 9x15bold
!*DisplayFont: 7x13bold
*TuneDisplay.geometry: 836x509+0+127
*Qtune.geometry: 297x509+845+127
*TuneCalibration.geometry: 318x156+824+665
*TuneHelp.geometry: 631x508+203+127
```

Figure 24. Example of `app-defaults` X Resource Setup File

Chapter 19. Other Networking Options

Sections in this chapter:

- 19.1 “Hardware Options,” this page
- 19.2 “Direct Connections,” page 262
- 19.3 “Using a Modem,” page 263
- 19.4 “Using Mail on a Client Machine,” page 264
- 19.5 “Avoiding Disk Full Problems with Mail,” page 264
- 19.6 “Other Software,” page 266

This chapter is an attempt to cover the broad topic of how to connect two Sun systems through a direct RS-232 serial link, a modem, a phone connection, or a packet-switched data network (PSDN). Also included are some instructions on using electronic mail (e-mail) on systems connected to Internet:

This topic is quite extensive, the software and its configuration files are rather convoluted and, to some extent, specific to the hardware involved. It is not possible to cover this topic completely in this manual. In many cases, it will be necessary for you to seek the help of an expert to solve site-specific problems.

19.1 Hardware Options

The simplest way of connecting two systems via serial line is a direct RS-232 connection. Compared to RS-232, Ethernet is about 100 times faster, simpler to use, more reliable, built into every Sun, and can span over longer distances (depending on the hardware). For these many reasons, the RS-232 connection is almost never used except for a special application, such as to provide some loose but safe connection between a mail server on a public network and another system on a local network, to prevent outside people from breaking into a local network. With such a connection, overall transfer rates of up to approximately 1750 bytes per second (at 19,200 baud) can be obtained (compared to several hundred kilobytes per second with Ethernet).

The main application for connections via serial lines is to link two systems that are not located in the same site. In fact, any two systems in the world can, in principle, be connected using the serial ports. There are two basic options for doing this: using regular phone lines in a dial-up connection or using a dedicated line into a public data network. Phone lines are typically accessed via modem. Common modem transfer rates are 9600 to 28800 baud (about 1000 to 3000 characters per second) or sometimes over 50 kilobaud using data compression.

To access public networks (assuming you cannot access the Internet via local network and gateway), a simple dial-up serial line using a modem is not good enough. You need a full IP (TCP/IP) connection such that e-mail transfers, FTP (data transfers with error checking), telnet (remote logins in general), X terminal operation over Internet, and Web browsing are possible. There are a number of alternatives:

- Direct networking connection to a local gateway computer
- PPP (or SLIP) over a modem
- PPP over ISDN
- X.25 over a dedicated line
- Leased line access

These alternatives are often accessed through a dedicated line. Special protocols using synchronous data transfer allow for transfer rates between 2400 baud and 64 kilobaud (about 6000 bytes per second). The line rental fee depends mostly on the data transfer rate. Alternatively, dial-up connections to the data networks are permitted (using normal modems, but mostly only at 1200 to 2400 baud, or about 100 to 200 characters per second). Usually, this precludes automatic mail delivery to the dial-in site.

Most universities are already hooked up to public networks, and users that are linked to the local university data network can also access public networks by communicating to a gateway computer via Ethernet. In these special situations, there are local experts who know how to establish a connection, and this case will not be covered here (except for the discussion of `mail`).

19.2 Direct Connections

Direct data transfer is possible without a software protocol, with the `tip` program, and using `uucp` on a direct line. Each is covered in the following sections.

Transfers without Software Protocol

Direct data transfer can be achieved without any special command or utility. Simply connect two Sun systems via one of their RS-232 ports such as `/dev/ttya` (port a), using a null modem (see "Connecting Serial Terminals," page 90). On the receiving system you type

```
cp /dev/ttya targetfile
```

This puts anything that comes from port a into a new file *`targetfile`*. On the transmitting system, you can now send the contents of that new file to the RS-232 connection:

```
cat sourcefile > /dev/ttya
```

This copies the file via RS-232. Because nothing has been specified, the transfer happens under standard conditions (9600 baud, 7 bits, no parity bit, 1 stop bit). The `stty` command can be used to change the speed if a different speed is required:

```
stty 1200
```

This command, which must be entered beforehand on both systems, changes the baud rate to 1200. The `echo` command automatically terminates the transmission by sending a Ctrl-d as terminating end-of-text (EOT) character, but on the receiving system, which is still waiting for further data, the file transfer must still be terminated by pressing Ctrl-c.

Using tip on a Direct Line

The above procedure provides only very rudimentary error checking, if any, and the integrity of the transferred data cannot be guaranteed. Also, it is rather awkward as a method, as it requires coordinated actions on both systems involved *for every file transfer.*

It is much better to use software such as `tip` that provides proper error checking, such as checksum comparison, and permits handling data transfers from one side only. `tip` is much more because it allows for a full-duplex terminal session on a remote host. Again, we connect two serial ports via a null modem. On the remote system, set up the corresponding port (`/dev/ttya` or `/dev/ttyb`) for terminal operation and send a hangup signal to the `init` process (see "Connecting Serial Terminals," page 90).

Note that the default speed on Sun serial ports is 9600 baud. You can use this value on both systems. The serial line serves as a one-way terminal connection, and you must not activate a login on the local port.

To define a port for `tip` on the local machine, edit the file `/etc/remote` by changing the entry `hardwire` for the correct port and baud rate (if necessary):

```
hardwire:\
        :dv=/dev/ttya:br#9600:el=^C^S^Q^U^D:ie=%$:oe=^D:
```

For an explanation of the entries, see the file `/etc/remote`. You can now use `tip`:

```
tip hardwire
```

At this point, you are in a terminal session, working on the remote system. You should get a login prompt and can work on the remote system. The login session is terminated by a entering `logout` or by pressing Ctrl-d.

To terminate the `tip` session (you don't log you out automatically), type a special escape sequence using the tilde (~) sign (other escape sequences serve to transfer files):

`~.`	Terminate `tip` session (no automatic `logout`)
`~p source <`*`target`*`>`	Send a file to the remote computer, and if *`target`* is not specified, it has the same name as the local file
`~t source <`*`target`*`>`	Receive a file from a remote computer, and if *`target`* is not specified, it has same name as the remote file
`~c <`*`directory`*`>`	Change directory on the local machine

Sessions using `tip` can also be preconfigured and customized with the file `~/.tiprc`. See Sun documentation for details. Note that `tip` can only transfer ASCII files, because control characters are used to terminate the file transfer.

Using uucp on a Direct Line

The `tip` program only allows for online file transfers during a login session, and only ASCII files can be transferred. The `uucp` (UNIX-to-UNIX copy) program allows off-line transfers and some binary file transfers. Using `uucp` on a direct line provides for a loose, but safer link, between a system with public network access and a local network.

19.3 Using a Modem

Not only is there a multitude of modems on the market, but each modem type has different properties and requires a different setup (a different software interface, in essence). Trying to cope with the different modem types on the market would be beyond the scope of this manual. You are referred to the documentation that accompanies the modem.

19.4 Using Mail on a Client Machine

A mail client machine is a system that only knows two classes of mail—local messages and all other messages. Mail between two local users is distributed directly, and all other mail is sent to a mail server for further distribution.

Mail administration on such a machine is fairly simple. In `/etc/hosts`, add the alias `mailhost` to the machine that is the mail server on your network, for example:

```
192.9.200.1      unity500 loghost
192.9.200.2      vxr400
192.9.200.3      vxr300
192.9.200.4      sun mailhost
127.0.0.1        localhost
```

You don't even have to reboot. Your system is now ready to send e-mail to any place and user the mail server can reach.

19.5 Avoiding Disk Full Problems with Mail

Receiving mail is a slow process because a network (TCP/IP over Ethernet or X.25, or `uucp` over X.25 or modem) is involved.

Receiving Large Mail Files

Incoming mail is first built up in the directory `/var/spool/mqueue` or in the directory `/var/spool/uucp/<`*`remotehostname`*`>` on systems using `uucp` for mail transfers. While mail is arriving, a typical listing of the `mqueue` directory may look as follows:

```
-rw-------  2 root            0 Aug  5 14:12 lfAA01781
-rw-------  2 root            0 Aug  5 14:12 qfAA01781
-rw-r--r--  1 root            0 Aug  5 14:12 xfAA01781
-rw-------  1 root            0 Aug  5 14:12 dfAA01781
```

The file `df<`*`message-ID`*`>` receives the incoming message and grows as the message arrives (in many cases, the directory listing does not indicate the intermediate or final size, because the directory is never adjusted before the file is deleted again). The other files in the directory (`lf<`*`message-ID`*`>`, `qf<`*`message-ID`*`>`, etc.) are described below.

After the entire message is received, it is then copied into the file `/var/spool/mail/<`*`username`*`>`. Temporarily, disk space of twice the size of the message is required (the source file is only deleted when the target file is completely written out). Later on, when looking at the message through `mailtool`, the `mailtool` program opens a temporary file `/tmp/MTda<`*`PID`*`>`, and again, disk space of twice the size of the message is required.

Sending Large Mail files

Building up an outgoing message is a fast process (several megabytes of disk space can be eaten up in a few seconds) because it is purely local. As a consequence, the disk may get filled very quickly, but you may not be able to see which file is using up the disk space, because the directory entry for that file is probably never really made (it would only show up after completing the file).

The only command that shows disk space usage is `df`. It turns out that for sending mail, the data is built up in `/tmp`. The message is built up in one file and then copied into a file `/var/spool/mqueue/df<`*`message-ID`*`>` such that, temporarily, the space requirement is twice the size of the message (concluding from the disk space consumption, it seems that before copying the data to `/var/spool/mqueue`, a second copy of the data is made temporarily within `/tmp`).

The other files in `/var/spool/mqueue` are used with the spooling software: `lf<`*`message-ID`*`>` and `xf<message-ID>` are lock files, and `qf<message-ID>` contains routing and header information for `sendmail` and `smtp`.

How to Avoid Problems with Mail

Solutions to disk full problems with mail include the following::

- *Increase the size of the root partition* – This is not really recommended because it is not trivial (it requires reformatting the entire disk and reloading UNIX) and it permanently takes away disk space from other partitions. Furthermore, with a smaller internal hard disk (e.g.: less than 500 MB), it is often impossible to increase the size of the root partition, because this limits the size of the swap space or the size of `/usr` (and hence the number of loadable software options) or both. See Chapter 2, "Software Installation," for a method for changing the size of the disk partitions during the Solaris installation.
- *Move the spooling directories to another disk partition* – This has the advantage that it doesn't require changing disk partitioning (you can work with standard disk partitions) and no additional free space is permanently locked away in the root partition. Moving spooling directories to another partition also resolves problems with editing large files, and it can easily be expanded to resolve problems with plotting large files from a remote system (so the data resides in `/var/spool/<`*`hostname`*`>_<`*`portname`*`>`, not in `/vnmr/tmp`). The following procedure assumes a large disk partition `/data` (with enough free space guaranteed) exists:

  ```
  su
  cd /
  tar cf - tmp var/spool/mail var/spool/mqueue |(cd /data; tar xvfBp -
  mv tmp tmp.std; ln -s /data/tmp tmp
  cd var/spool
  mv mail mail.std; ln -s /data/var/spool/mail mail
  mv mqueue mqueue.std; ln -s /data/var/spool/mqueue mqueue
  ```

  It is obvious that filling `/data` can now disable mail spooling. It should be made sure that this partition is never filled to 100%.

  Systems using `uucp` should also include `/var/spool/uucp`:

  ```
  su
  cd /
  tar cf - tmp var/spool/mail var/spool/mqueue var/spool/uucp | \
      (cd /data; tar xvfBp -)
  mv tmp tmp.std; ln -s /data/tmp tmp
  cd var/spool
  mv mail mail.std; ln -s /data/var/spool/mail mail
  mv mqueue mqueue.std; ln -s /data/var/spool/mqueue mqueue
  mv uucp uucp.std; ln -s /data/var/spool/uucp uucp
  ```

A more general solution to spooling disk space problems would be the following:

```
su
cd /
tar cf - tmp var/spool | (cd /data; tar xvfBp -)
mv tmp tmp.std; ln -s /data/tmp tmp
cd var
mv spool spool.std; ln -s /data/var/spool spool
```

Remember, this is not a 100% safe solution. The directory `/data` can also fill up, which then can lead to problems receiving and sending mail, printing and plotting, and even editing files. On the other hand, this may be the only solution that permits sending or receiving large data files.

19.6 Other Software

If data should be transferred to a PC that is not equipped with Ethernet, this can also be achieved through serial line (RS-232) using the Kermit protocol. For Sun computers running VNMR software, the Kermit software has been submitted to the user library (for Sun-4 computers only). No further discussion of this subject is provided here.

For Apple Macintosh computers, a software package, PlotView (from Stevens Creek Software, 21346 Rumford Drive, Cupertino, CA 95014), allows capturing plotter output directly through RS-232 (the Macintosh is hooked up as a plotter). This software is made specifically for HP-GL plotter output and is designed for direct incorporation of spectral results into reports, publications, etc.

Another application of RS-232 communication is the use of terminals and terminal emulation software on IBM PCs and Apple Macintoshes. This (as well as the use of X terminals and terminal emulators) is not discussed any further here, mainly because in this case the processing happens on the workstation, and only display information is transferred to the terminal.

Chapter 20. Solaris 2.x vs. SunOS 4.1.x

The difference between SunOS 4.1.x (BSD UNIX) and Solaris 2.x (AT&T SVR4 UNIX) are considerable. Trying to cover all this in a chapter of this manual would be a pointless exercise. However, there are still many users still running SunOS 4.1.x that soon will upgrade to Solaris 2.x or to a new workstation running Solaris 2.x. For most users, the transition is fairly easy, especially given the new CDE graphical user interface. On the other hand, migrating shell scripts to Solaris 2.x can often cause problems and extra work, and that is where we want to provide help in a future version of this manual:

- To point out critical areas for shell scripts where things change, and which may require special attention, and
- To indicate how you could write shell scripts compatible with *both* SunOS 4.1.x *and* Solaris 2.x.

With these hints, the transition to Solaris can be made almost painless, reducing the disruption in your productivity to a minimum.

Chapter 21. Calibration Tests and Shimming

Sections in this chapter:

- 21.1 “Calibrating Pulse Width Manually,” this page
- 21.2 “Testing Decoupler Field Strength,” page 270
- 21.3 “Testing Decoupler 90° Pulse Width with Polarization Transfer,” page 271
- 21.4 “Calibrating ^{1}H Decoupler Pulse Width with PPCAL,” page 273
- 21.5 “Calibrating ^{13}C (or X) Decoupler Pulse Width with PWXCAL,” page 274
- 21.6 “Shimming,” page 275
- 21.7 “AutoTest—Automated Instrument Testing,” page 284

This chapter describes calibration tests and shimming procedures for a Varian NMR spectrometer system. Tests described here cover pulse width calibration, decoupler field strength, decoupler 90° pulse width, and decoupler pulse calibration. There are two ways to perform calibrations: manually or by automatic calibration. As described in the manual *Walkup NMR Using GLIDE*, calibration can be placed in a single probe file, saving the user the problem of updating numerous parameter sets with the same numbers.

21.1 Calibrating Pulse Width Manually

Sample: Any sample;
Parameter set: None.

1. Set up parameters to obtain a spectrum on the sample of interest.
2. Using either the macro `movetof` or the macro `movesw`, arrange the spectral window so that at least one resonance falls relatively near the center of the spectral window. So that you can repeat any experiments using intervals between pulses several times greater than that T_1, it is helpful to estimate the relaxation time T_1 of the sample.
3. Using a pulse that you know (or suspect) to be less than 90°, do one transient (`nt=1`) with no steady-state (`ss=0`) and absolute intensity mode (`ai`). Phase that spectrum properly.
4. Estimate the 90° pulse width, multiply by 4 to get the 360° pulse width, and enter an array around the 360° pulse width.
5. Make sure `d1` is greater than 3 times T_1, and acquire data using `ga`.

   The signals should be negative if the pulse is shorter than 360°, zero if the pulse is 360°, and positive if the pulse is longer than 360°.
6. Select a value of `pw` that gives the result nearest to zero, using rough mental interpolation if none of your results were exactly zero. Divide this value by 4 to give the 90° pulse width and enter this value in `pw90` as well as in your log book. To be

sure you were not off by a factor of two, set `pw` equal to a 180° pulse and obtain a spectrum—the result should be near zero. Now set `pw` to a 90° pulse and the result should be a maximum.

In many cases, the peak is never *exactly* zero but instead shows a “dispersive” signal with some signal positive and some negative. Do *not* readjust the phase in this case. This is normal behavior. Simply select the value of `pw` that gives “equally balanced” up and down resonances.

21.2 Testing Decoupler Field Strength

Sample: 60% C_6D_6/40% dioxane (5-mm probe, Part No. 00-968120-69;
10-mm probe, Part No. 00-968123-69; 16-mm probe, Part No. 00-949134-69)
Parameter set: `/vnmr/tests/gamah2`

The strength of the *decoupler field,* known as γH_2, is important to know for a number of reasons:

- `dmm='f'` (swept fm or fm-fm modulation) decoupling, the decoupler field strength gives a rough measure of the range over which protons will be efficiently decoupled. Thus at 200 MHz, one might want a 10 ppm or 2 kHz decoupler field.; at 300 MHz, a 3 kHz decoupler field; etc.
- `dmm='w'` (WALTZ-16) decoupling, protons are efficiently decoupled over roughly twice that range; that is, efficient decoupling over a 2 kHz range can be achieved using only a 1 kHz decoupler field strength.
- For WALTZ-16 decoupling, the decoupler field strength must be known because the modulation frequency parameter `dmf` must be set to equal $4 \bullet \gamma H_2$.
- Various experiments that require the use of decoupler pulses will also require a knowledge of the decoupler field strength.

Decoupler field strength is a function of the decoupler power level (controlled by the parameters `dpwr` or `dhp`/`dlp`) and the probe. To a lesser extent, but especially when using highly ionic samples that can “detune” the probe, the decoupler field strength also depends on the sample. For “normal” organic solvents, it is usually sufficient to calibrate the decoupler field strength for each probe at a variety of settings and perhaps to repeat the calibration every few months. For polar solvents, and samples in water at high buffer concentrations, it may become necessary to calibrate the decoupler on the sample of interest, or at least on a comparable sample.

The standard method of calibrating the strength of the decoupler field is off-resonance decoupling. Two experiments are performed, one with the decoupler at a higher frequency than the proper decoupling frequency for a particular proton, and one with the decoupler at a lower frequency. This technique produces two carbon spectra with “reduced” couplings—multiplets that have basically the same pattern as in a coupled spectrum (doublets, triplets, etc.) but in which the coupling constant is reduced. With these two spectra plus a knowledge of the full coupling constant, an appropriate equation can be used (see K.G.R. Pachler, *J. Magn. Reson.* **7**:442 (1972)) to determine the value of γH_2. The following procedure is recommended:

1. Insert the standard ^{13}C sensitivity sample (60% C_6D_6 and 40% dioxane).

   The dioxane produces a single resonance, a triplet with 1:2:1 amplitude, when coupled (from the CH_2 carbon in dioxane). This pattern will change when decoupling is used.

2. Retrieve an appropriate parameter set (assuming that `dpwr` or `dhp` is set correctly) by entering rtp('/vnmr/tests/gamah2')
3. Acquire two spectra by entering ga
4. Display the first spectrum with two cursors by entering `ds(1)`
5. Position the cursors on the outer lines of the triplet, as shown in Figure 25.
6. Read the value `delta` from the screen. Divide the result by 2 (because we really want just the value of a single splitting) and write down that number.
7. Display the second spectrum by entering `ds(2)`
8. Position the cursors in a similar fashion, and read the value of `delta`. Again write down half that difference.

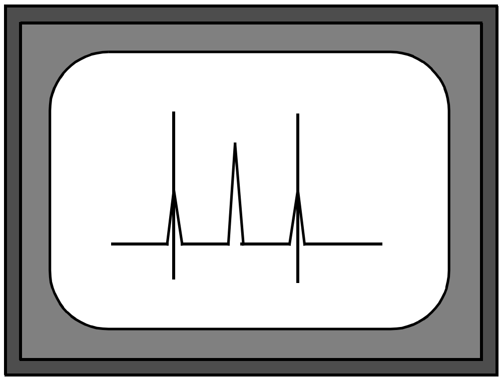


Figure 25. Measuring Residual Couplings in Dioxane

9. Start the program to calculate the strength of the decoupler field by entering: `h2cal`.

   When the system prompts for the low-field residual coupling value, enter the result from step 6.

   When the system prompts for the high-field residual coupling, enter the result from step 8.

   When the system prompts for the full coupling constant, enter `142`, the value for dioxane.

   The system displays the calculated value of the decoupler field strength γH_2, the predicted coalescence point (the frequency at which single-frequency decoupling would collapse the dioxane to a singlet), and the pulse width for decoupler pulses if this decoupler level is to be used for pulsed decoupling.

21.3 Testing Decoupler 90° Pulse Width with Polarization Transfer

Sample: 30% menthol in $CDCl_3$ (Part No. 00-968120-94)
Parameter set: `/vnmr/stdpar/C13`

The decoupler 90° pulse width value (parameter `pp`) from the test above should be appropriate for polarization transfer experiments. It is also possible to calibrate `pp` separately by using a polarization transfer pulse sequence directly, on the sample of interest. As an example of this procedure, a sample such as menthol having CH, CH_2 and CH_3 carbons is useful. If menthol is unavailable, use any small molecule organic compound that is highly soluble.

1. Enter **`jexp2 setup('C13','CDCl3') nt=4 ga`**.
2. Phase the spectrum and place the two cursors around the aliphatic region.
3. Enter **`movesw ga`** to narrow the spectral window. Phase the new, narrowed spectrum.

4. Enter **`jexp1 setup('H1','CDCl3') nt=4 ga`**.
5. Place the cursor in the center of the aliphatic region and enter **`sd`**. Note the value of `dof` (decoupler offset) after entering `sd`.
6. Enter **`jexp2`** and set **`dof`** to the value just found by `sd`.

   As an alternative to this step, set the decoupler offset `dof` to –2.5 times the spectrometer frequency of the system (e.g., on a 200-MHz system, set `dof=-500`; on a 400-MHz, set `dof=-1000`).
7. Enter **`dept`**. After the help file is displayed, enter an estimate of the decoupler 90° pulse width that was obtained previously, or some other conservative estimate.
8. Enter **`mult=0.5 d1=1 ss=2 nt=16 ga`**.
9. Phase the data when finished. All resonances should be positive.
10. Enter **`pp=10,20,30,40,50,60,70`** and rerun the experiment.
11. After the data are transformed, enter **`dssh`**.
12. Select the value of **`pp`** that gives the maximum peak heights (use `da` to check the array).
13. Enter a new array of **`pp`** values, bracketing the value determined in step 12 by ±20%, using about six smoothly-spaced values. Rerun the experiment and determine the `pp` value for maximum intensity, the same as step 12. This value is a reasonable estimate of the decoupler 90° pulse width.

For spectral editing with the `adept` program or MAGICAL macros such as `cdept`, `hcdept`, or `dept`, you must have a very accurate value of the decoupler 90° pulse width. Obtain this by observing `pp` dependence of CH_3 and CH_2 carbons for `mult=1.0` (CH selection).

1. Enter **`mult=1`**.
2. Make an array of **`pp`** in steps of 1 µs, from –5 to +5 µs around the `pp` value determined above.
3. Enter **`ga`** and when finished, enter **`dssh`**.
4. With the CH carbons phased vertically, the CH_2 carbons should go from positive to negative with increasing `pp`. The CH_3 carbons decrease to zero and then increase. The `pp` value corresponding to a decoupler 90° pulse is the value that nulls the CH_2s. Ideally, the J value used in the parameter set should correspond to that for the type of CH_2 in the molecule: 125 Hz for menthol. If a more accurate value is needed, reset the `pp` range to cover a smaller span and make the steps in `pp` smaller.
5. Update the system log book.

21.4 Calibrating ^{1}H Decoupler Pulse Width with PPCAL

The PPCAL pulse sequence is used to calibrate the proton decoupler pulse width for experiments such as DEPT, INEPT, and HETCOR. Figure 26 shows the sequence.

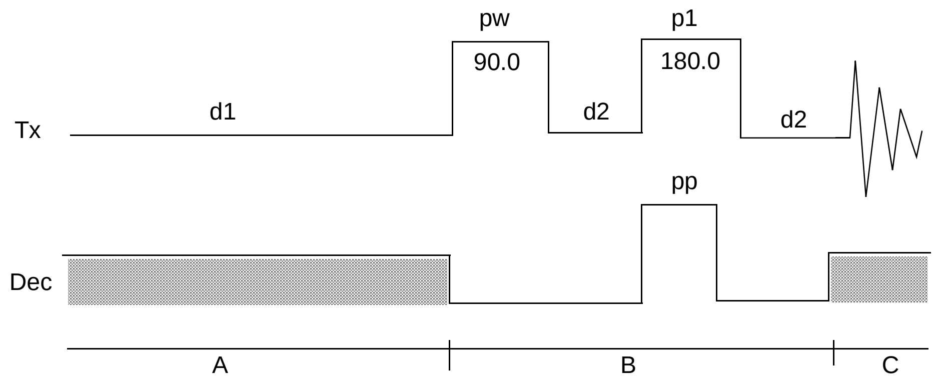


Figure 26. PPCAL Pulse Sequence

Parameters

The `ppcal` macro sets up the parameters for PPCAL as follows:

`pp` is a proton 90° decoupler pulse (in µs).

`d2` is a delay that should equal 1/(2*J_{CH}) (in sec).

`pw` is a 90° pulse on ^{13}C (in µs).

`p1` is a 180° pulse on ^{13}C (in µs).

`dm` is the decoupler mode. The value should be `'yny'`.

`dmm` is the decoupler modulation mode. Its value should be `'wcw'` or `'fcf'`.

`pplvl` is the power level for the proton decoupler pulse. `dpwr` is the power level for broadband proton decoupling if the decoupler channel uses a linear amplifier. If the decoupler channel uses a class C amplifier, maximum power is used for the proton decoupler pulse and `dhp` is the power level for broadband proton decoupling.

Technique

The following technique is recommended:

1. Array parameter `pp`, starting at 0. Make sure that delay `d1` is reasonably long compared to the ^{13}C relaxation times.
2. Phase the first spectrum (`pp=0`). CH and CH_3 carbons should go from positive to negative, and CH_2 from positive to zero and again positive. All peaks should null when `pp` is a 90° pulse. The CH carbons are the most sensitive.

21.5 Calibrating ^{13}C (or X) Decoupler Pulse Width with PWXCAL

The PWXCAL pulse sequence (not supplied with *GEMINI 2000*) is used to calibrate the pulse width characteristics of the probe's decoupler channel(s) in indirect detection or triple resonance experiments. PWXCAL can also be used to determine the rf field homogeneity of the decoupler. This calibration is a more sensitive measure of the decoupler X pulse widths than the first increment of HMQC. PWXCAL is designed for dual-broadband systems only and does not support "reverse mode" acquisition.

Parameters

The `pwxcal` macro sets up the parameters for PWXCAL as follows:

`pw` is a proton pulse width (in µs)

`pwx1` is a 90° pulse on X (in µs) for the first decoupler.

`pwx2` is a 90° pulse on X (in µs) for the second decoupler.

`pwx3` is a 90° pulse on X (in µs) for the third decoupler.

`jC13`, `jN15`, or `jP31` should be set to the appropriate coupling constant.

`jname` is set by the `pwxcal` macro to indicate which nucleus has been selected.

`dm` is the decoupler modulation and must be `'nnn'`.

`dmm` is the decoupler modulation mode and should be `'ccc'`.

`dof` is the X-nucleus resonance location (note: CH_3I `dof` is `–14800` for a 500-MHz system at 11.4 T).

`dpwr` (or `dpwr2`) is the power level of the "X" decoupler pulse, and `tpwr` is the power level for proton observe if the decoupler channel uses a linear amplifier.

Technique

The following technique is recommended:

1. Starting from a 1D proton parameter set, type `pwxcal` and answer the questions "use decoupler 1, 2, or 3 [1]" and "calibrate C^{13}, N^{15}, or P^{31} [C^{13}]". Pressing the Return key to either of the questions selects the default response enclosed in square brackets.
2. Array parameter `pwx1`, starting at 0, making sure that delay `d1` is reasonably long compared to the ^{1}H relaxation times.
3. Phase the first spectrum (`pwx1=0`). All peaks null when `pwx1` is a 90° pulse.
4. If a second decoupler is present, the parameter `pwx2` is arrayed to calibrate the 90° pulse width on that decoupler. If a third decoupler is present, the parameter `pwx3` is arrayed to calibrate the 90° pulse width on that decoupler.

21.6 Shimming

Shims are a set of coils inside the magnet that induce changes in the shape of the magnetic field. Each shim produces a specific change in the magnetic field that can be easily shown. To provide a visual reference for the interactions of the shims, the approximate shapes of the axial gradients (spinning shims) are shown in Figure 27.

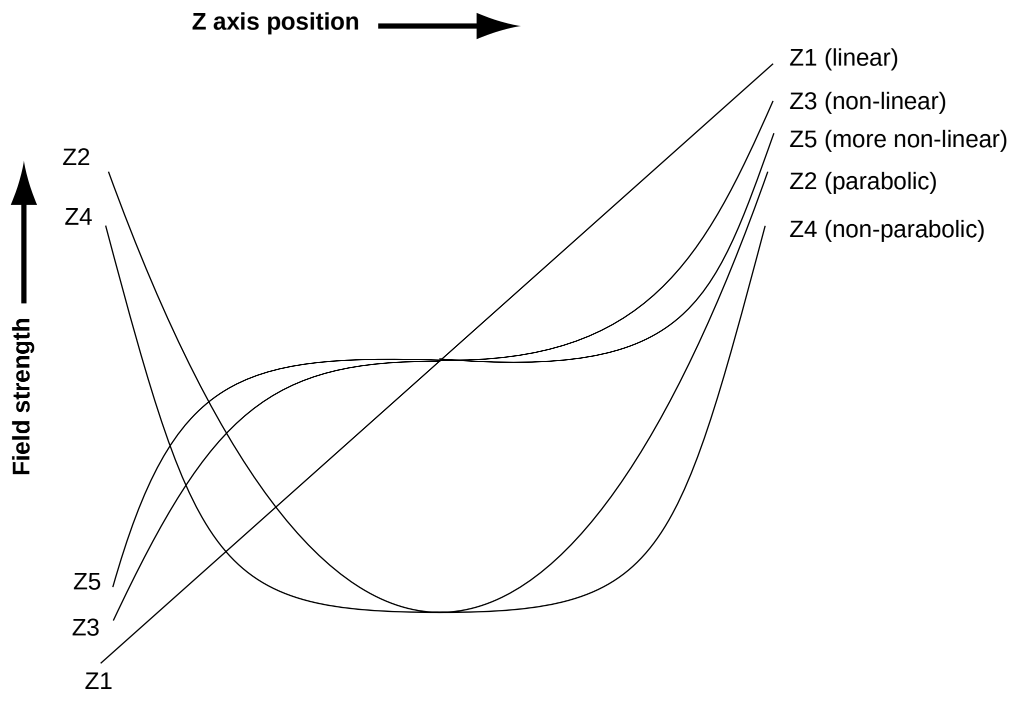


Figure 27. Approximate Shape of Axial Gradients

Understanding the effect of various shims on symmetry of the resonance is important in simplifying the shimming process. The following two points must be considered:

- The effect of a given shim on the spectral lineshape.
- How the shims interact with each other.

Understanding how the shims interact is critical to simplifying the task of shimming. Pure shim gradients produce a very specific and predictable effect on the magnetic field and, to a lesser extent, on the resonance lineshape.

For gradient shimming, refer to the manual *User Guide: Liquids NMR*.

Shim Interactions

The following sections show theoretically predicted changes in lineshape caused by changes in shim DAC values. Shim sets with pure shims, such as the Varian Ultra•nmr shims, follow the theoretically predicted response very closely. Other shim systems, with more interactions between shims, produce somewhat different results.

Theoretically Perfect Lineshape and Effect of Z1 Shim

Figure 28 shows a theoretically perfect line shape (at left) produced in a perfectly homogeneous field (at right). The magnetic field shape appears as a flat line, indicating that the magnetic field does not change across the length of the sample.

Figure 29 shows how changing the linear shim Z1 affects the line shape and the magnetic field.

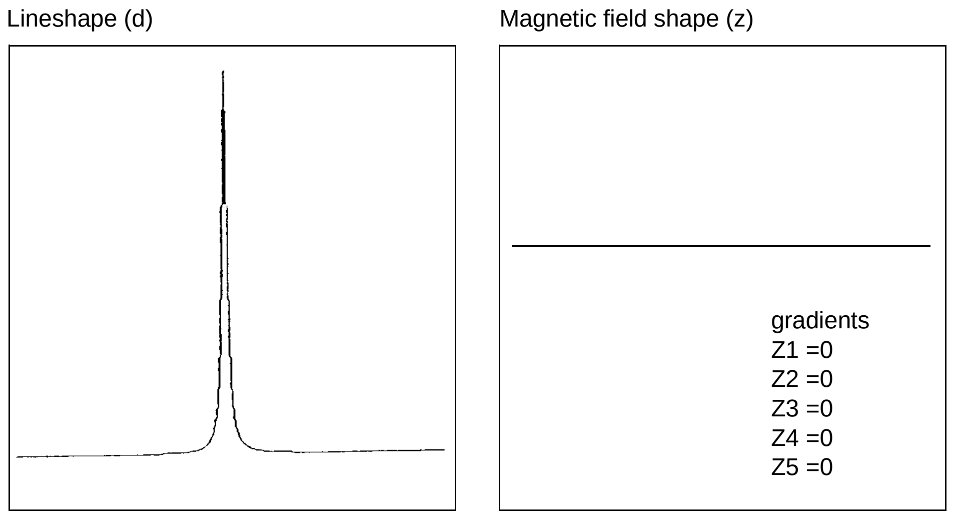


Figure 28. Theoretically Perfect Lineshape

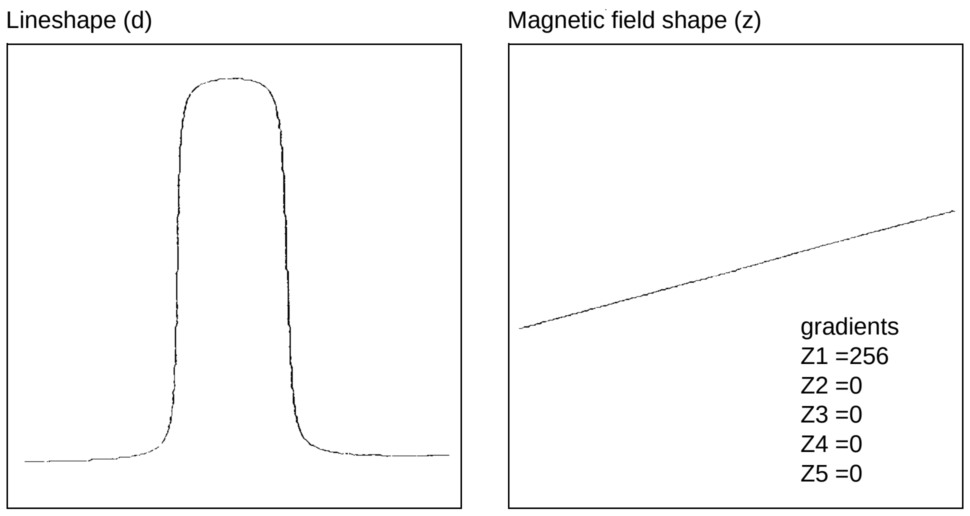


Figure 29. Effects of Linear Shim Z1

Effects of Even-Order Shims Z2 and Z4

Figure 30 shows the effect of the even-order shims, Z2 and Z4, on the line shape. Notice that a positive misadjustment of both shims produces an upfield tail on the peak. If Z2 and Z4 are misadjusted in the negative direction, the asymmetry occurs on the downfield side of the peak. The difference between Z2 and Z4 is in the height of the asymmetry. The Z2 shim causes asymmetry higher on the peak than Z4.

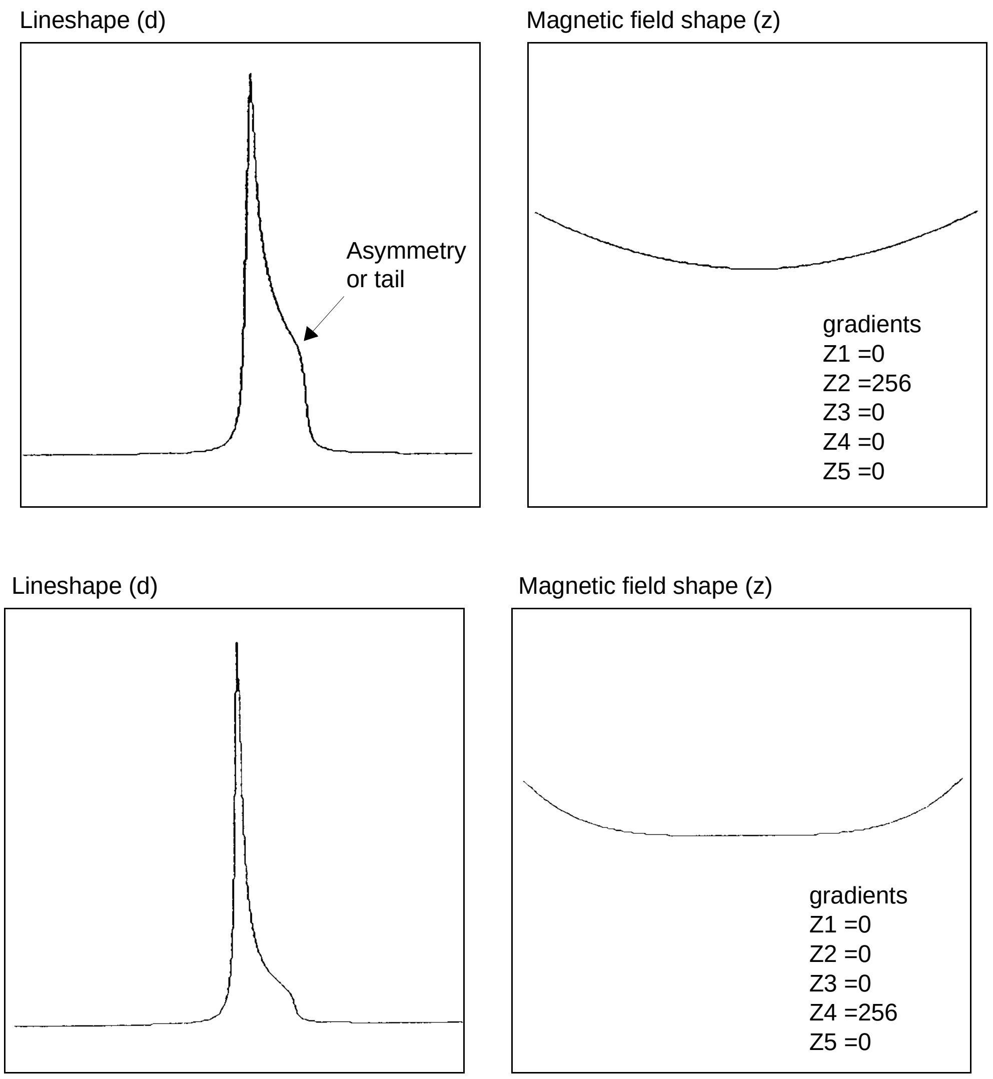


Figure 30. Effects of Even Order (Parabolic) Shims Z2 and Z4

Effects of Odd-Order Shims Z3 and Z5

Figure 31 shows the effects of the odd-order shims Z3 and Z5 on the line shape. The odd-order shims cause broadening of the peak and therefore affect resolution.The Z5 shim is unavailable on systems with 13-channel shim sets (`shimset`=1).

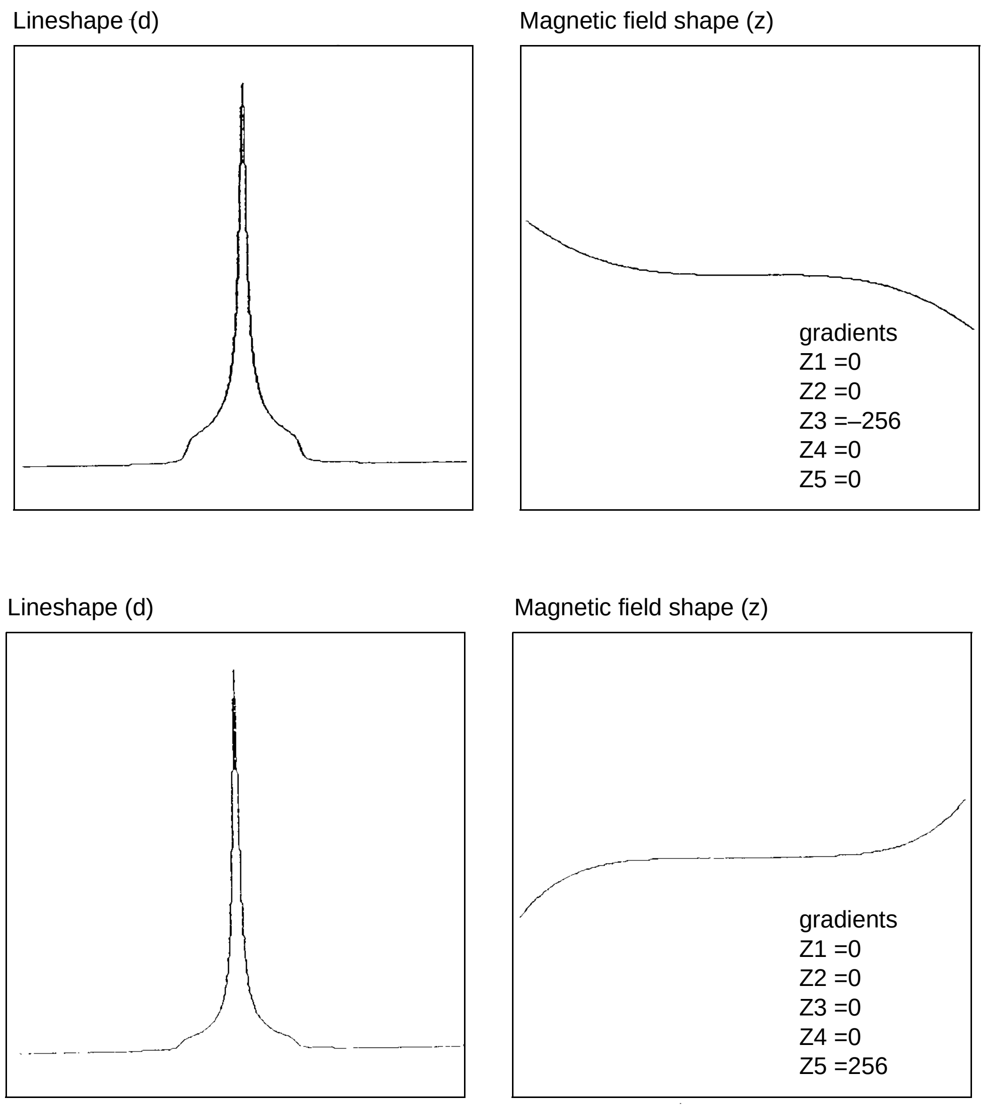


Figure 31. Effect of Odd Order (Non-Linear) Shims Z3 and Z5

Effects of Misadjusted Shims

Figure 32 shows two examples of the effects when more than one shim is misadjusted. This is the typical case with real samples. The complex line shapes make simple visual analysis difficult. A procedure for correcting the shims is provided later in this section that can be used as a guide when adjusting shims.

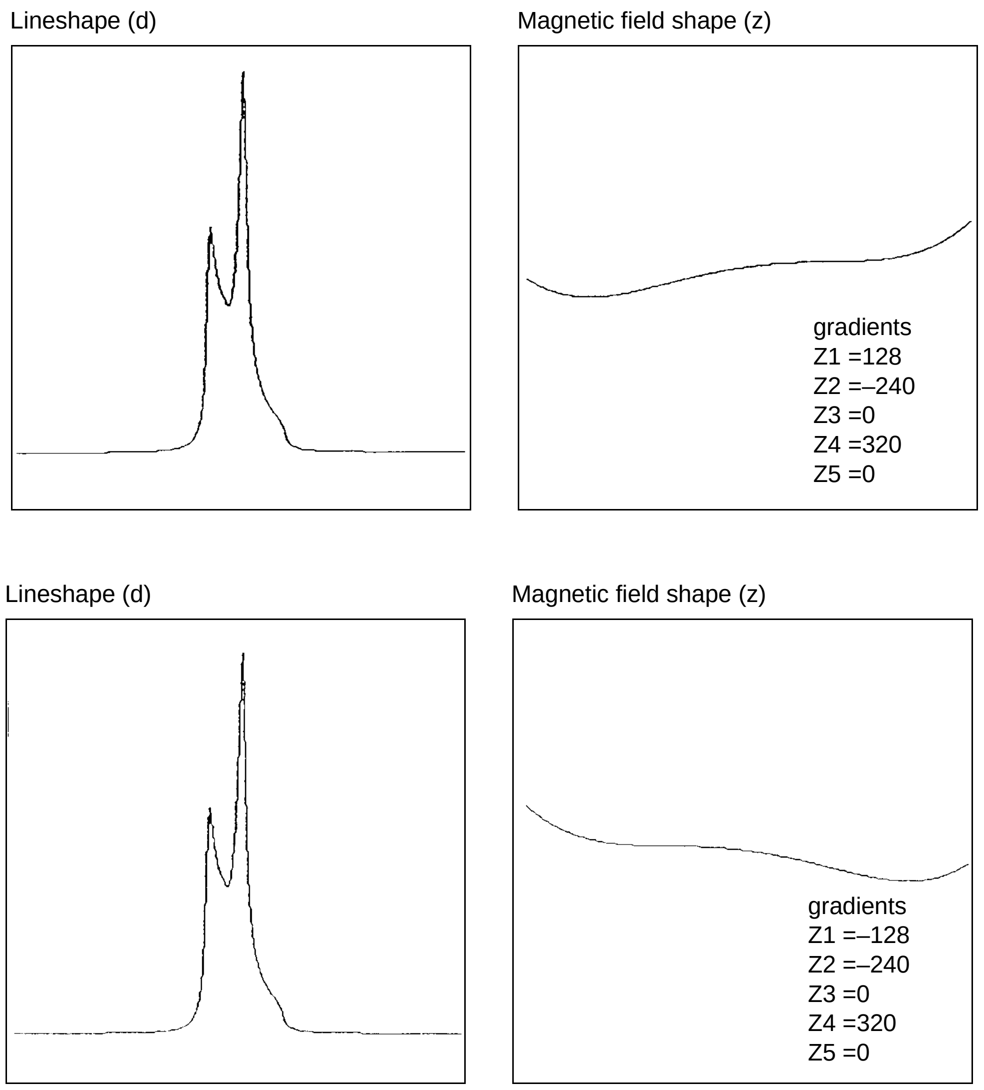


Figure 32. Effects of Misadjusted Shims

Effects of Non-Spin Shims

Figure 33 shows the effect of the non-spin shims on the spectrum (note that Z3X and Z3Y are not available on 13- or 14-channel shim systems). If set wrong, the first-order non-spin shims (X, Y, ZX, and ZY) can cause first-order spinning sidebands. XY and X2–Y2 can cause second-order spinning sidebands. High-order non-spin shims can cause a broad peak base.

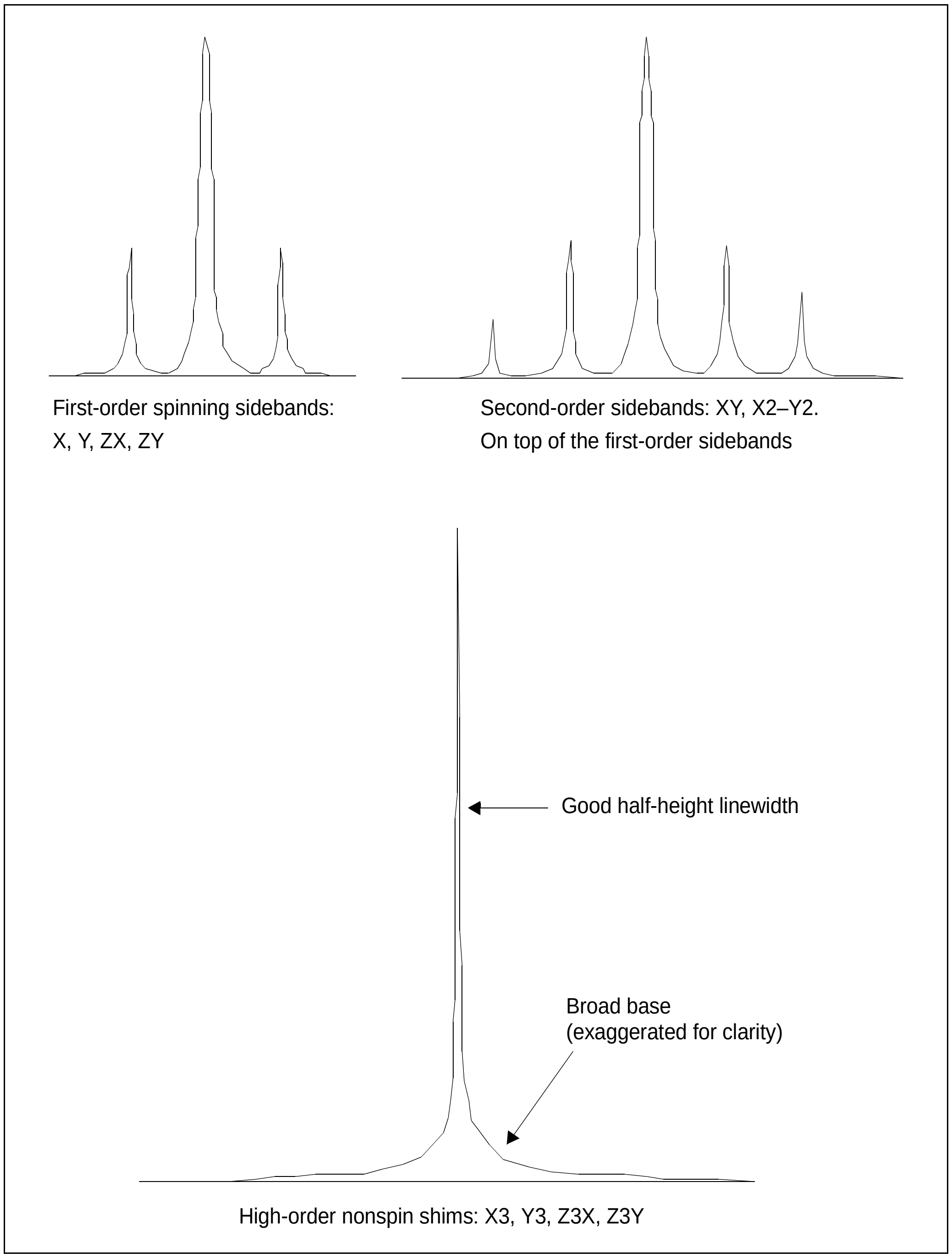

First-order spinning sidebands:
X, Y, ZX, ZY
Second-order sidebands: XY, X2–Y2.
On top of the first-order sidebands
Good half-height linewidth
Broad base
(exaggerated for clarity)
High-order nonspin shims: X3, Y3, Z3X, Z3Y

Figure 33. Effects of Nonspin Shims

Summary of Shim Interactions

Table 14 lists some line shape effects associated with shims. Note that 13-channel shim systems (`shimset=1`) do not have Z5, Z3X, ZXY, etc., and that 14-channel shim systems (`shimset=10`) have Z5 but do not have Z3X, ZXY, etc.

Table 14. Lineshape Effects and Their Associated Shims

Lineshape Effect	*Shims*
Split peak	Z4 and Z1
Asymmetry greater than half-way up	Z2
Asymmetric foot	Z4
Symmetric feet and or low broad base	Z5
Symmetrically broad base	Z3
Spinning sidebands	Low-order radials X1, Y1
Symmetric broad base	High-order radials X3, Y3, etc.

Typical interactions for axial shims:

- Z1 and all other axial shims, to some extent
- Z2 and Z1
- Z3 and Z1
- Z4 and Z2 (with large delta Z4s: Z4 and Z3)
- Z5 and both Z3 and Z1 (Z5 not available on 13-channel shim systems)

Setting Low-Order (Routine) Shims

The following procedure describes how to set the low-order, or routine, shims. You may need to reset Z0 and lock phase if you are making very large changes in the room temperature shims. With this procedure, you should concentrate on improving the symmetry of the main resonance as well as the half-height resonance and line shape.

1. Click on the **Connect** button in the Acquisition window.
2. Click on the **SHIM** button and set SPIN to on
3. Adjust the lock level to about 80 if possible.

   Maximize lock level with Z1.

   Maximize lock level with Z1 and Z2. Do this by making a change in Z2 followed by maximizing with Z1 again. Continue to iterate in this manner until you can no longer increase the lock level.
4. Acquire the spectrum.

   If the sample is properly shimmed, the lines should be symmetric.
5. If the lines are not symmetric or unusually broad at the base, refer to Table 14 and the previous sections for which shims to adjust. You should not need to adjust Z3, Z4, or the non-spins for most routine samples.
6. If you do need to adjust Z3, do so by interactively shimming Z1 and Z3 in the manner described in step 3 for Z1 and Z2. Changes in Z3 may affect Z2 so after shimming Z3 maximize Z1 and Z2 again.

Removing Spinning Sidebands (Non-Routine)

If the spinning sidebands are not within specification, use this procedure to remove them.

1. Write down the lock level, set SPIN to off, and write down the lock level.
2. Adjust lock to about 80 if possible.
3. Maximize lock level with X.
4. Maximize lock level with Y.
5. Maximize lock level with X and Y.

   Do this by making a change in Y followed by maximizing with X again. Continue to iterate in this manner until you can no longer increase the lock level.
6. Maximize lock level with X and ZX.

   Do this by making a change in ZX followed by maximizing with X again. Continue to iterate in this manner until you can no longer increase the lock level.
7. Maximize lock level with Y and ZY.

   Do this by making a change in ZY followed by maximizing with Y again. Continue to iterate in this manner until you can no longer increase the lock level.
8. Repeat step 4 above.
9. Maximize lock level with XY and ZXY (ZXY not available on 13 or 14 channel shim systems).
10. Repeat step 4, step 5, and step 6.
11. Set SPIN to on and acquire a spectrum.

    If the sample is properly shimmed, the lines should be symmetric.
12. If the lines are not symmetric or unusually broad at the base, refer to Table 14 and the previous sections for which shims to adjust. For most routine samples, you should not need to adjust Z3, Z4, or the non-spins.
13. If you need to adjust Z3, do so by interactively shimming Z1 and Z3 in the manner described in step 3 in the previous procedure (“Setting Low-Order (Routine) Shims”) for Z1 and Z2.

    Changes in Z3 may affect Z2 so after shimming Z3 maximize Z1 and Z2 again.

Setting the High-Order Axial Shims (Non-Routine)

1. If Z4 needs to be adjusted, look at which side of the peak the asymmetry appears—low field to the left and high field to the right.
2. Use Figure 30 to determine which direction to move Z4. If the asymmetry is large (Z4 is far off), change Z4 by a considerable amount to try to push the asymmetry to the other side of the peak. This provides two important pieces of information:
   - Confirms that Z4 is the problem if the asymmetry moves.
   - Indicates what the actual value of Z4 should be when Z4 is changed. Since you know the values that caused it to be on either side of the peak, the correct value must be between the two extremes.
3. Set Z4 to the value that produces neither a high-field nor low-field asymmetry.

   Z4 affects all the shims below it, so repeat the in the “Setting Low-Order (Routine) Shims” procedure.

4. Maximize the lock level with Z5.
5. Repeat step 3 and step 4 until no further increase is obtained.

Setting High-Order Radial Shims (Non-Routine)

Note that Z2X, Z2Y, ZX2–ZY2, Z3X, Z3Y, and Z5 are not available on 13-channel shim systems.

1. Set SPIN to off and write down the new lock level.
2. Set the lock level to about 80.
3. Maximize the lock level by shimming Z2X against ZX.
4. Maximize the lock level by shimming Z2Y against ZY.
5. Repeat the “Removing Spinning Sidebands (Non Routine)” procedure.
6. Maximize the lock level by shimming ZXY against XY.
7. Maximize the lock level by shimming ZX2–ZY2 against (X2-Y2).
8. Set SPIN to on and adjust the lock level to 80.
9. Maximize the lock level by shimming Z1, Z2, Z4, and then Z1, Z2, Z3.
10. Repeat step 1.
11. Maximize the lock level by shimming X3 against Y3
12. Maximize the lock level by shimming Z3X against Z3Y if available.

    Refer the Oxford manual for your magnet for approximate Z3X and Z3Y values. Be aware that the signs may be reversed in the Oxford manual, so you will have to experiment to determine the correct sign.
13. Look at the spectrum and decide where to concentrate your effort:
    - For a broad base, adjust Z4 and Z5.
    - For spinning sidebands, adjust the proper order radial shims.

    As Z4 and Z5 are optimized, the contribution of Z3 to the breadth of the base becomes more clear, as well as any contribution from the high-order radial shims. Several cycles of shimming are required.

    In some cases, local maxima will be encountered, causing the greatest problems. A local maxima may be indicated if a high-order shim continues to increase and eventually reaches the maximum output of the shim supply, without having reached the optimal lock level.

    In such a case, carefully reexamine the lower-order shims by making large excursions (systematically), beginning with the lowest-order shim and working up. This is a particularly difficult issue when dealing with the high-order radial shims such as X3, Y3, Z3X, and Z3Y, because their perturbation of the lock level is small relative to the change in the shim current.

    At the same time their perturbation of the spectrum is significant in experiments such as water suppression, but their effects can go unnoticed or may not be important in some routine 1D spectra, where large solvent peaks are not encountered.

Note that Z-axis gradient shimming can be used on UNITY*INOVA*, UNITY*plus*, UNITY, and VXR-S systems. Refer to the manual *User Guide: Liquids NMR* for instructions.

21.7 AutoTest—Automated Instrument Testing

AutoTest is a series of automated tests for the console and the probe. These tests demonstrate the console performance and its compliance with the acceptance test specifications.

The tests automatically write the results to a text file and optionally plot the resulting spectra (although plotting is optional, it should be activated for acceptance testing and customer review). At the time of system installation, a hard copy of the AutoTest report is attached to the appropriate *Acceptance Test Results* form.

WARNING: **Remove any files used for previous versions of AutoTest from your `vnmrsys` directories, particularly any `~/vnmrsys/seqlib/AT*` or `~/vnmrsys/maclib/AT*` files. Remove `AT*.DEC`, `gauss32.RF`, `gauss.RF` and `eburp1.RF` from `~/vnmrsys/shapelib` if present. Make sure that `~/vnmrsys/maclib/autotest` is not present.**

AutoTest Step-by-Step Procedure

The following procedure provides an overview on running AutoTest. Subsequent sections cover configuring and running the software, as well as the experiment involved, in detail.

1. Make sure all elements of the rf system (transmitters, linear modulators, rf attenuators, amplifiers, receivers, and probes) are in the standard configuration.
2. If the system has a PFG (pulsed field gradients) accessory installed, make sure that the gradient amplifier is on and **`pfgon`** is set correctly for the number of gradients available.

   •For VNMR, set `pfgon='nny'` for Z-axis only or `pfgon='yyy'` for triax.

   •For VNMRJ, use the **utilities** drop-down menu and select **`system settings`** to set the active gradients.
   Allow sufficient time for stabilization. AutoTest calibrates the gradients.
3. Use the sample described in AutoTest Sample on page 285.
4. Set the VT to 25° C. Allow the temperature of the sample to regulate and equilibrate. While specifications are determined at 25° C, normal day-to-day AutoTest runs may be done at other temperatures.
5. Tune the probe, and lock on the D_2O resonance.
6. Shim the field on the sample to give a *nonspinning half-height that is dominated by the paramagnetic relaxation agent. This should require little time since the H_2O line is quite broad*.
7. Note the **`tpwr`** value necessary to produce a ^{1}H 90° pulse width that is within specification (8 to 10 µs) but does not cause probe arcing. AutoTest uses this `tpwr` value and determines the ^{1}H 90° pulse width and calculates the amplifier compression at this `tpwr` value.
8. If you want to save the results of a previous AutoTest run, rename the `history` and `data` directories.
9. For VNMR, enter the macro **`autotest`**. For VNMRJ, use the **utilities** drop-down menu and select the AutoTest option from the calibrations sub-menu. In the AutoTest display, specify the options and enter values for power and an approximate ^{1}H `PW90` for this value of `tpwr` (10 µsec recommended). Specify the coil size (in mm). This value is 18 mm for most Unibody-style (hexagonal base) probes and

16 mm for all others. A full AutoTest run including all available options must be run before any single test or partial set of tests is specified. Most of these tests rely on calibrations that are performed as part of the full AutoTest run. This option is specified by the All Standard Tests checkbox.

After selecting the **Begin Test** button, AutoTest begins. The total time for the test(s) will depend on the test(s) specified and on plotting, and CPU speed.

As AutoTest runs, the FIDs are stored in the `data` directory, and the results from the tests are stored in the `history` directory.

AutoTest Sample

As the sample, AutoTest uses 0.1% ^{13}C-enriched methanol in 1% H_2O/99% D_2O. The sample is doped with gadolinium chloride at a concentration of 0.30 mg/ml, which produces a ^{1}H T_1 relaxation time of about 50 to 75 ms. The resulting line width is considerably larger than the magnet-determined line width because of the paramagnetic relaxation contribution, minimizing any dependence on shimming skill and permitting rapid collection of data.

Starting AutoTest

AutoTest is started by entering the macro **`autotest`** or by selecting a menu button that runs this macro. The system first checks for an `autotest` directory in the user's current `vnmrsys` directory. If no `autotest` directory is present, `~/vnmrsys/autotest` is automatically created and subdirectories are copied from `/vnmr/autotest`. These directories include parameter, data, history and database `atdb` directories.

The AutoTest program appears similar to Figure 34. If needed, this window can be resized using the mouse. The Begin Test button should be visible in the lower left corner. If not, resize the window.

Across the top of the window, several *tabs* appear, including Configuration, Test Library, History, and Test Report. Selecting one of these changes the display:

AutoTest Window

Configuration Tab

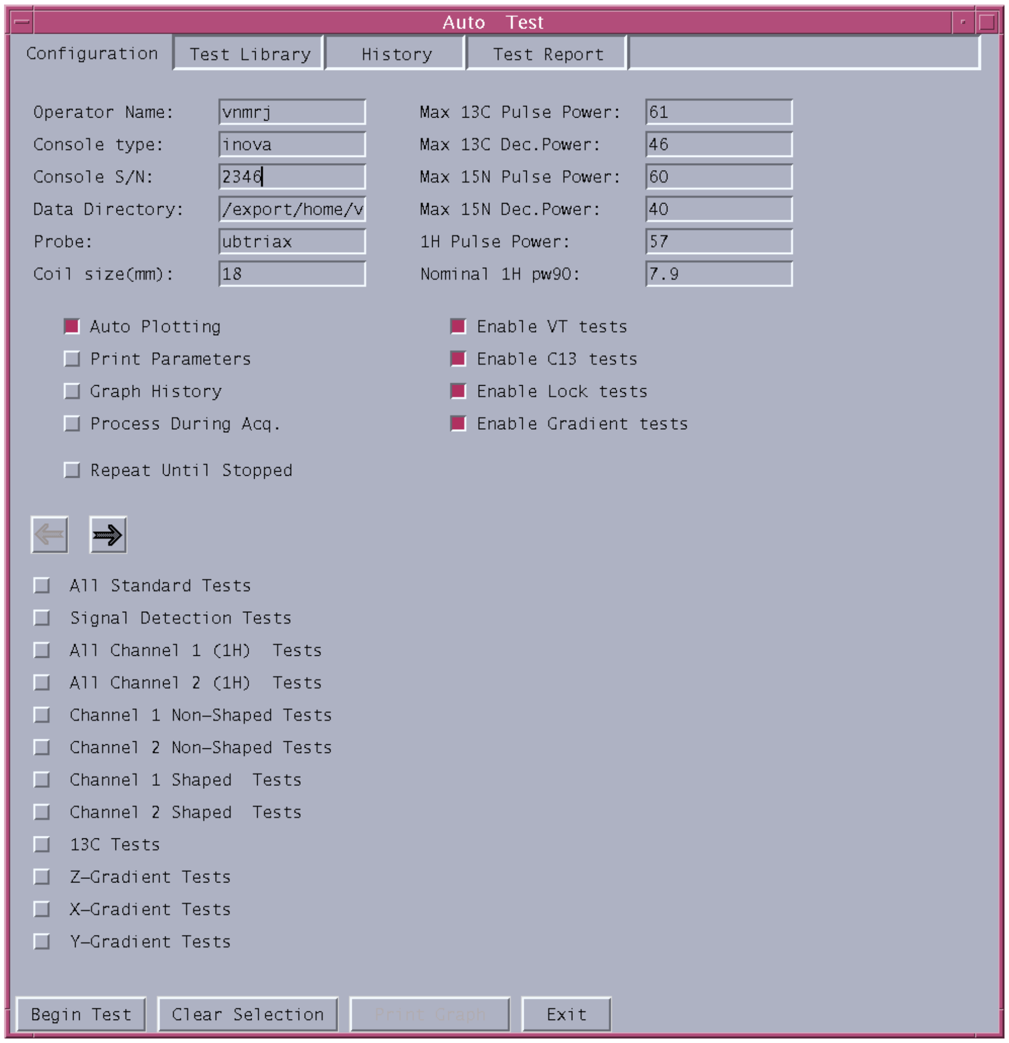
The Configuration tab selects a display with entry fields and checkboxes for configuring AutoTest.

Entering System Information

At the top of the Configuration display are fields for entering user, console, and probe identification as well as the nominal `tpwr` and `pw90` values for ^{1}H.

The `tpwr` and `pw90` values are used as starting points and should be entered the first time AutoTest is run. The ^{1}H pw90 value should be correct to within ±30%. The power level entered is used for most of the tests so it should be a level that is used in normal day-to-day work. For most probes this would be a power level sufficient to get a 10 µs `pw90`.

Running at the upper range of `tpwr` may result in a situation where the amplifier is in compression. This means that although the `pw90` may be shorter at `tpwr=63` than `tpwr=57`, the `pw90` at the former power may not be a factor of two shorter. If a 100-watt high-frequency amplifier is present, the `tpwr` value should be set so that ^{1}H pw90 is 10 µs



Operator Name:	vnmrj	Max 13C Pulse Power:	61
Console type:	inova	Max 13C Dec.Power:	46
Console S/N:	2346	Max 15N Pulse Power:	60
Data Directory:	/export/home/v	Max 15N Dec.Power:	40
Probe:	ubtriax	1H Pulse Power:	57
Coil size(mm):	18	Nominal 1H pw90:	7.9

- [✓] Auto Plotting
- [ ] Print Parameters
- [ ] Graph History
- [ ] Process During Acq.
- [ ] Repeat Until Stopped
- [✓] Enable VT tests
- [✓] Enable C13 tests
- [✓] Enable Lock tests
- [✓] Enable Gradient tests

- [ ] All Standard Tests
- [ ] Signal Detection Tests
- [ ] All Channel 1 (1H) Tests
- [ ] All Channel 2 (1H) Tests
- [ ] Channel 1 Non-Shaped Tests
- [ ] Channel 2 Non-Shaped Tests
- [ ] Channel 1 Shaped Tests
- [ ] Channel 2 Shaped Tests
- [ ] 13C Tests
- [ ] Z-Gradient Tests
- [ ] X-Gradient Tests
- [ ] Y-Gradient Tests

Figure 34. AutoTest Program Display

at that power. In any case, the amplifier compression at the entered `tpwr` is determined, with the assumption that `tpwr`-12 dB is a setting where there is linear behavior.

The length (in mm) of the active window in the receiver coil is also specified. This is normally 16 mm, unless the probe has a hexagonal base (Unibody-style), in which case the coil may have an 18 mm length (normally true for indirect and triple-resonance liquids probes).

Some probes require more careful power control (very high field and cryogenic). Power control for ^{13}C and ^{15}N pulses and decoupling is available within this panel. These values are used to set upper limits on power (attenuator) settings for ^{13}C and ^{15}N 9 (using channels 2 and 3) pulses and decoupling for those tests which have power limits. These are primarily decoupling noise tests. The ^{13}C pw90 calibration is normally done for a power level giving a nominal 15 µsec pulse and this power level is found by experiment. This power value is limited by the above values. If the probe used has no ^{15}N port, set the maximum pulse power to zero. The relevant macros check for a zero value and skip ^{15}N pulses or decoupling for this case. AutoTest macros including `c` after `AT` have power limits internally coded.

Configuring Tests

The top-left column of checkboxes in the AutoTest window selects test configurations:

- Auto Plotting – Automatic plotting of spectra after each experiment. This box should be on the first time AutoTest is run to produce a hardcopy record, but can be off for subsequent normal maintenance mode.
- Print Parameters – Automatic plotting of a separate page that includes a parameter set, pulse sequence and descriptive text (only if Auto Plotting is selected). This box should be on the first time AutoTest is run to produce a hardcopy record, but can be off for subsequent normal maintenance mode.
- Graph History – Automatic plotting of history graphs (only if Auto Plotting is selected).
- Process During Acq. – Automatic (`wnt`) processing and display of spectra after each FID.
- Repeat Until Stopped – Automatic repeating of AutoTest (until manually aborted).

These checkboxes reflect the setting (`y` or `n`) of global parameters `at_plotauto`, `at_plotparams`, `at_graphs`, `at_wntproc` and `at_cycletest`, respectively. These global parameters, as well as those mentioned for test selection, are updated when the Begin Test button is selected.

In most cases, a single AutoTest run is appropriate and the Repeat Until Stopped checkbox should not be set. For troubleshooting, or to establish a statistically valid database, however, you might want to run the test overnight. When the Repeat Until Stopped checkbox is selected, the automatic processing and display of data after each FID option is disabled after the first pass (because it is unlikely that the user is present to view the data). In addition, all function tests that do not produce a numerical result are skipped.

Note: If you select Auto Plotting, Print Parameters, or Graph History, make sure enough paper is available for the printer or plotter.

Enabling Tests

The top-right column of checkboxes on the AutoTest window set whether VT, C^{13}, lock, and gradient tests are enabled or not. These check boxes should be set appropriately before selecting the Begin Test button. If these are not selected, any tests involving them are skipped in an AutoTest run, even if the tests are selected in a test package or individually.

The normal use of these check boxes is to allow skipping of some tests when the All Standard Tests package is used. The boxes reflect the current state of the options when the AutoTest program starts (these boxes reflect the user global parameters values, `y` or `n`, of `at_vttest`, at_`c13tests`, `at_locktests`, and `at_gradtests`, respectively).

Selecting Test Packages

In the center of the AutoTest window is a column of check boxes used to select test packages.

The first checkbox is All Standard Tests. This option sets AutoTest to make a full run, checking out all the relevant hardware enabled by the upper checkboxes.

All Standard Tests must be the first test performed (with all options enabled) because it determines calibrations for any of the specific tests. Once the full AutoTest is run, any single test may be run and it uses the calibrations stored in the standard parameter set or in the global variables updated by specific tests. Thus, a single test can be run without doing any calibrations, but its accuracy is dependent on the last calibration performed.

As AutoTest experiments are completed, relevant calibrations are stored in special global parameters. In addition, the `standard.par` parameter set (stored in `autotestdir+/parameters`) is updated whenever the `tof`, `pw90`, `T1`, linewidth, or gain values are determined. These are determined in the first few experiments of the All Standard Tests run, or if specific tests are requested.

Thus, if only a single specific test is requested, the `standard.par` parameters should have appropriate values so that a full autocalibration of all parameters is unnecessary. The relevant global parameters include C^{13}-related parameters, gradient calibrations, etc. The macro `/vnmr/maclib/maclib.autotest/ATglobal` creates these variables (if not already present) and you can read it to get an idea of the global variables used by AutoTest. For VnmrJ, the **utilities**, menu has AutoTest information showing the values of most of the AutoTest global parameters.

Below the All Standard Tests checkbox are several checkboxes for different packages of tests. By selecting one or more of these, you can chose the tests performed. Selections may be unselected by clicking the checkbox once again.

If there is not enough room within the display to show all choices, a pair of arrow symbols appears at the top of the list. Selecting the right arrow updates the display with the next list of choices. Selecting the left arrow returns to the previous page of choices.

Test Library Tab

Selecting the Test Library tab at the top of the AutoTest window displays a list of specific tests. Use the right and left arrow buttons to view groups of individual tests organized by hardware, for example, channel 1 tests, z-axis gradient tests, C^{13} tests, etc.

Selection of one or more of the checkboxes in any of these displays specifies the tests to be performed, in the order selected. Tests may be *unselected* by using the checkbox once again. Any of these tests may be grouped with the group test(s) specified in the Configuration panel.

Experiments are initiated by returning to selecting the Configuration tab and clicking the Begin Test button.

History Tab

Selecting the History tab at the top of the AutoTest window provides graphical output of the `history` files accumulated after several AutoTest runs.

To select a `history` file, scroll to the name of the file. The left arrow and right arrow may be used to rapidly step through the list.

Buttons are provided to specify a graphical or text output of the history file. If more than one result is contained in the history file, a menu of choices is displayed to the right of the display of results.

Values of results that exceed specified limits are displayed in red in the graphical output and in color in the text output. If a history file has multiple results with specifications, all displayed points for that run will indicate failure if only one has failed. Thus, a particular display may show a failure even though the result is within specification. If this is true, select the other entries in the list to show which result had actually failed.

Printed graphs may be obtained by selecting the button at the bottom of the screen. A full set of small graphs is plotted automatically after an AutoTest run when the Graph History checkbox is selected in the Configuration display (if automatic plotting is requested).

Test Report Tab

Selecting the Test Report tab at the top of the AutoTest window allows viewing the `atrecord_report` file, a report on the results for the current AutoTest run. The `atrecord_report` file is one of two files that contain test results (both files are stored in the `autotestdir` directory):

- The `REPORT` file is more compact and is automatically printed at the end of the run.
- The `atrecord_report` has the same results, but in a format that is similar to the `history` file format. In addition, it indicates a Fail status if any of the results in a history file line is out of bounds relative to the upper and lower limits stored in the `atdb/at_specs_table` file.

The `atrecord_report` report may be scrolled to view all of the results. All failures are indicated. Tests that have passed defined upper and lower limits, and tests for which there are no defined upper and lower limits have no Pass/Fail status indicated. A rapid indication of any failures is given by selecting the Fail Only option. Either mode of the report can be printed by selecting the Print Report button at the bottom of the display.

Failed experimental data is stored, with a date-stamp, in a `data.failed` directory which is created in the `autotest` directory (if needed). If tests are repeated, new date-stamped files are created at each failure, permitting later examination of the failed experiments. In addition, a file FAILREPORT is stored in the `autotestdir` directory and it is printed out at the same time as the REPORT file. This report shows the `history` file entry for all failures and the time of the failure. This report is also stored with date-stamp in the `autotestdir reports` directory when a new AutoTest run is begun.

Saving Data and FID Files from Previous Runs

As AutoTest executes, data and FID files are written into the `history` and `data` directories, which are located in the `autotest` directory. The `autotest` directory is usually located in the directory `vnmrsys` of a user's home directory. The contents of the `data` directory are progressively overwritten as AutoTest continues.

Before starting a new AutoTest run, do the following to save the data from a previous run:

1. Open a UNIX window. and enter **`cd ~/vnmrsys/autotest`**.
2. Change the name of the `history` directory by entering, for example,
   `mv history history.old.`
3. Change the name of the `data` directory by entering, for example,
   `mv data data.old`

Creating Probe-Specific Files

If you run AutoTest with different probes, you should keep separate `autotest` directories. Use the following steps to create probe-specific files.

1. After you have run AutoTest using a specific probe, change the name of the `autotest` directory by using the `mv` command, for example:
   ```
   cd ~/vnmrsys
   mv autotest autotest_probe_1
   ```
   Where *`probe_1`* is the name of the probe that was tested, for example, `5mmTriplePFG` or `5mmID`.

   Any new AutoTest run automatically creates a new `autotest` directory in the user's `vnmrsys` directory. The only file that needs to be updated would be

~/vnmrsys/autotest/parameters/standard.par. This can either be copied from the saved `autotest` file or the parameter set may be retrieved using `rt` or `rtp`, the parameters updated and then saved, replacing the `standard.par` file. This should be safe for any parameters displayed in the `dg` window, but there are several parameters dealing with gradients and indirect detection that must also be checked. It is safest to do an All Standard Tests run the first time a new probe is used.

Once a calibrated `standard.par` parameter set is present, `autotest` directories may be renamed whenever a probe is changed. In this way, `history` files may be kept specific to a probe.

2. To change the file name back to `probe_1` (or the name you have chosen), enter, for example:

```
cd ~vnmrsys
mv autotest autotest_probe_2
```

Where *probe_2* is the name you have chosen for the probe last tested.

```
mv autotest_probe_1 autotest
```

Where *probe_1* is the name you have chosen for the probe you now want to test.

If you need to repeat any individual test, you can do so by recalling the appropriate FID from the `data` directory. The experiment can then be started with the `go` command without overwriting the previous data. Or the test may be selected from the Test Library after using the `autotest` macro or a menu calling this macro.

AutoTest Directory Structure

AutoTest uses the `~/vnmrsys/autotest` directories listed in Table 15.

Table 15. AutoTest Directories.

Directory	*Contents*
`data`	FIDs from the recent AutoTest run(s)
`data.failed`	FIDs from any failed Auto Test experiments
`history`	History files for the various tests
`reports`	Copies of the report generated each time AutoTest is run
`parameters`	Parameter files—default entry is `standard.par`
`texts`	Copies of the text files attached to the AutoTest experiments
`atdb`	AutoTest database

data Directory

The `~/vnmrsys/autotest/data` directory contains FIDs collected in previous AutoTest experiments. As each experiment finishes, the macro specified by the `wexp` parameter executes, and as part of that macro, a `svf` command is performed that saves the FID under a file name specified by the parameter `at_currenttest` (if it contains a name). The macro first removes any file by the same name (the results of the test from the last time it was run) and then executes `svf`. Thus, the data directory may contain FIDs obtained during different AutoTest runs if those runs were not full runs.

Any data files stored in the data directory can be recalled by normal VNMR commands such as `rt`. The data may then be transformed and displayed. The `wexp` parameter will contain the name of the macro normally used for data processing, so that the `wexp` command can be used to duplicate the actions normally done in an automatic manner.

Note: **If file `~/vnmrsys/autotest/atdb/at_selected_tests` is empty, only processing and no further acquisition is done. If the file `~/vnmrsys/autotest/atdb/at_cycled_tests` is not empty, those tests may start. Therefore, clear the contents of at_selected_tests and `at_cycled_tests` before manually executing the macro.**

This result is normally the case if the last AutoTest run came to a normal completion. However, if the last AutoTest run was aborted and no new entry into the AutoTest Program was done, this file will contain entries and an acquisition may start up following the `wexp` command. If so, just abort the acquisition.

data.failed Directory

The `~/vnmrsys/autotest/data.failed` directory contains any data from any failed experiment. Failure results when a calculated result falls outside limits defined in the `~/vnmrsys/autotest/atdb/at_spec_table` file. Varian specifications are indicated in the `~/vnmrsys/autotest/atdb/at_spec_table` file. Users can modify this file by supplying upper and lower limits. Any user-modified `at_spec_table` file should be saved outside `~/vnmrsys/autotest`, since this file can be deleted later.

parameters Directory

The `~/vnmrsys/autotest/parameters` directory contains any parameter set used by AutoTest macros, including any put there by the user. Normally, only `standard.par` is present. This parameter set has all parameters necessary for the AutoTest macros. Values of parameters may be displayed by using `dg` in the text window. Some parameters are only displayed when certain variables are nonzero, or `'y'` if a string parameter; however, these parameters are printed and displayed if used in an experiment. The AutoTest macro `Atrtp` is used to recall a parameter set from this directory.

reports Directory

The `~/vnmrsys/autotest/reports` directory contains text files from previous AutoTest runs, by date. Each run produces a report, whether plotting is requested or not. The report file for a currently proceeding AutoTest run is `~/vnmrsys/autotest/REPORT`. At the end of an AutoTest run, this file is copied to the reports directory under a title that includes the date and time. If AutoTest is repeated, automatically a new report is written out for each complete AutoTest run. The existing `~/vnmrsys/autotest/REPORT` file is renamed as `~/vnmrsys/autotest/LASTREPORT` whenever an AutoTest run begins. Similar actions are executed for the `atrecord_report`. This directory also stores any FAILREPORTs with appropriate date-stamps.

texts Directory

The `~/vnmrsys/autotest/texts` directory contains mainly text files that are printed on some spectral plots and most parameter set printouts. These files explain the purpose of the test.

history Directory

The ~/vnmrsys/autotest/history directory contains text files that record the values determined in AutoTest runs. They are generated automatically by the ATrecord macro which is used in any AutoTest macro that obtains a numerical result from an NMR experiment. Each result is written on a new line and is date-stamped. Tests that have a Varian specification listed in the manual *Acceptance Test Procedures* will be denoted as having passed or failed.

If the history file has more than one result per line, any one failure will cause a fail result for the whole line. When the history file is viewed using the History display (after using the macro autotest), failure is indicated by a red data point in graphical output and a colored entry in the text output.

If a user writes a new AutoTest macro including the ATrecord macro, the at_spec_table must be updated for the history files to be displayed. In addition, the new macro must be listed within the at_tests_file.

atdb Directory

The ~/vnmrsys/autotest/atdb directory contains mainly the following text files used by the Auto Test program to create the AutoTest interface:

at_tests_file

The at_tests_file file defines all the tests that AutoTest can perform. Tests are specified by a macro name and description. Normally, these are grouped and separated by a line starting with Label. The word following will be displayed as a heading for a group of tests. The test descriptions are displayed in the Test Library display (after entering the macro autotest or by using a menu calling this macro). The macro names are not shown in the display, just checkboxes next to the test description.

New tests may be added to the at_tests_file by specifying a group title (use the Label keyword as the first word on the line, followed by a descriptive phrase). Specify a macro name and then a test description, one per line.

at_groups_file

The at_groups_file file defines test packages that have been assembled for convenience. Each package has a line that gives a description (in double quotes) followed by a list of macros to be used in the order of acquisition. There are no restrictions on the placement of these macros in the text file, only the order matters. When the next double-quoted entry appears, a new group is set.

The AutoTest interface display shows these packages as checkbox entries in the Configuration display. Selection of one or more of these causes their execution in the order of selection, once the Begin Test button is selected. When this happens, the at_selected_tests file is fixed. Selection of the All Standard Tests checkbox disables any other selections that will be done as part of the All Standard Tests run.

Users may add new packages to the Configuration display list by adding appropriate lines to the at_groups_file in the same format.

Any macro specified within a group must be defined in the at_tests_file.

at_selected_tests

The at_selected_tests file contains the names of the macros to be run as part of the AutoTest procedure and is fixed at the time the Begin Test button is selected. The format is one line per macro with each line containing the name of a macro, in the order of acquisition.

As AutoTest proceeds, each line is deleted as the specified macro finishes its activity. Thus, completion of the AutoTest run is defined as when this file is empty. The single exception is the case of automatic repeating of AutoTest, as specified by the Repeat Until Stopped checkbox in the Configuration display and as indicated by the value of the global parameter `at_cycletest('y')`. In this case, at the completion of the AutoTest run, the contents of the file `at_cycled_tests` are copied into `at_selected_tests` and the process then continues.

at_cycled_tests

The `at_cycled_tests` file is updated when the Begin Test button is selected. If the Repeat Until Stopped checkbox is selected in the Configuration display, the global parameter `at_cycletests` is set to `'y'` and the file `at_selected_tests` is copied to `at_cycled_tests`. If no test cycling is requested, this file is emptied.

at_spec_table

The `at_spec_table` file is written out when the `~/vnmrsys/autotest` directory is created and is spectrometer-dependent. The appropriate file is copied from the directory `/vnmr/autotest`, depending on spectrometer frequency. It contains a list of macros used in AutoTest for producing entries in the history files. For each macro the following is specified:

- The history file affected by the macro.
- The column number (not counting date) containing the result.
- The lower limit for the result.
- The upper limit for the result.
- A text description of the result. This text description is used for the graphical displays and plots. A comment line above each macro serves to describe the test.

All results specified in the manual *Acceptance Test Procedures* have upper and lower limits specified numerically in this file. Those not having Varian specifications have asterisks (*) as entries for upper and lower limits and these results will have no indication of pass or fail in their history files, or colored indication of failure in the graphical displays of the history files.

Users may wish to set their own upper and lower limits for many, if not all, of the results. They may do so by replacing the asterisks with numbers. Of course, this should only be done after a good statistical base is obtained, such as more than 20 complete AutoTest runs. Once this base is obtained, the numbers put into the `at_spec_table file` should have a reasonable margin of error built in.

It is a good idea to make a copy of the file `at_specs_table file` prior to changing it, as well as the modified file, because deletion or renaming of the `autotest` directory will result in a default `at_spec_table` being copied from `/vnmr/autotest/atdb`.

AutoTest Macros

To help users who want to add or modify tests, this section describes some of the macros used by AutoTest. These macros are in `/vnmr/maclib/maclib.autotest`.

ATglobal Macro

The ATglobal macro is run when the AutoTest program begins. The macro checks for the existence of autotest parameters in the user file ~/vnmrsys/global. These parameters are used to store calibrations and results that are used by autotest macros. If the parameters are not present, ATglobal creates them. Otherwise, the parameters are left unchanged. In VNMRJ, the **Utilities** drop-down menu **system settings** option permits convenient viewing (and entry of) AutoTest global parameters. A partial list of these parameters is given in Table .Selected Parameters Created by ATglobal

Table 16. Selected Parameters Created by ATglobal

Parameter	*Contains*
at_currenttest	Name under which the FID is stored
autotestdir	Full path of the autotest directory
at_user	Name of the user running autotest (printed in report)
at_coilsize	length (in mm) of active window in coil (typically 16 or 18 mm)
at_consoletype	Name of console entered in AutoTest window
at_consolesn	Number of console entered in AutoTest window
at_probetype	Name entered for probe used in AutoTest window
at_wntproc	y or n (for processing/display after each FID)
at_cycletest	y or n (for automatic repeating of AutoTest)
at_printparams	y or n (for parameter list/pulse sequence printouts)
at_plotauto	y or n (for automatic plotting)
at_graphhist	y or n (for history graphs plotting)
at_locktests	y or n (for lock power/gain tests)
at_max_pwxlvl	Maximum permitted ^{13}C power level
at_max_pwx2lvl	Maximum permitted ^{15}N power level
at_max_dpwr	Maximum permitted ^{13}C decoupling power
at_max_dpwr2	Maximum permitted ^{15}N decoupling power
at_T1	Value of last determined T_1
at_gain	Value of gain determined by autogain
at_tof	Value of tof for water
at_fsq	Value of fsq parameter
at_dsp	Current value of dsp at start of run
at_ampl_compr	Value of high-band amplifier compression
at_LBampl_compr	Value of low-band amplifier compression
at_LBampl_compr_10usec_c	^{13}C amp compression at at_pwx90lvl_10µsec_c
at_decHeating	Temperature increase from decoupling
at_linewidth	Linewidth of water resonance
at_pw90	90° pw at power specified in AutoTest display
at_tpwr	Power specified in AutoTest display
at_pwx90c	^{13}C pw90 at power at_pwx90lvlc
at_pwx90lvlc	^{13}C power level for ~15-µs pwx90 at power<= to at_max_pwxlvl
at_pw90Lowpower	90° pw at reduced power
at_pwx90Lowpowerc	^{13}C pw90 at power at_pwx90Lowpowerlvlc
at_tpwrLowpower	Power level at reduced power

Table 16. Selected Parameters Created by ATglobal

Parameter	*Contains*
`at_pw90_ch2`	90° `pw` on channel 2
`at_pwx90`	^{13}C `pw90` determined at `at_pwx90lvl`
`at_pwx90lvl`	^{13}C power level for approximately 15-µs pwx90
`at_pwx90Lowpower`	^{13}C `pw90` at reduced power
`at_pwx90Lowpowerlvl`	^{13}C power level at reduced power
`at_pwx90_10usec_c`	^{13}C pw90 at power level at_pwx90lvl_10µsec_c
`at_pwx90lvl_10usec_c`	^{13}C power level for ~10-µs pwx90 at power<= to at_max_pwxlvl(typically at 800 MHz)
`at_pwx90Lowpower_10usec_c`	^{13}C pw90 at at_pwx90Lowpowerlvl_10µsec_c
`at_pwx90Lowpowerlvl_10usec_c`	^{13}C power level for at_pwx90Lowpower_10usec_c
`at_vttest`	y or n (for VT test)
`at_temp`	Current temperature
`at_vttype`	Current value of global parameter `vttype`
`at_tempcontrol`	Value reflects usage of `temp tcl/tk` panel
`at_gradtests`	y or n (for gradient tests)
`at_pfgon`	Current value of `pfgon`
`at_gmap`	y or n (for gradient mapping/shimming)
`at_gzcal`	Value of G/cm per DAC unit for z-axis gradient
`at_gxcal`	Value of G/cm per DAC unit for x-axis gradient
`at_gycal`	Value of G/cm per DAC unit for y-axis gradient

ATstart Macro

The `ATstart` macro is run after the Begin Test button is selected in the Configuration display. The macro sets the global parameters to reflect the current state of the hardware and aborts under certain circumstances, such as if requested tests are not compatible with the current hardware settings.

Messages are displayed indicating the source of the problem. The reports are initialized with relevant information and the ATnext macro is executed.

ATnext Macro

The `ATnext` macro checks the `at_selected_tests` file and copies the first entry into the global parameter `at_cur_smacro`, deletes the top line in `at_selected_tests` and executes the macro specified by `at_cur_smacro`. If the `at_selected_tests` file is empty, `ATnext` either finishes the AutoTest run or calls the `ATrestart` macro which copies the `at_cycled_tests` file to `at_selected_tests`, permitting repeated AutoTest runs until manually aborted by the user.

`ATnext` is usually found at the bottom of each macro defining a particular test. This permits the linking of one test to another, in a general fashion.

ATxxx Macros

Specific tests usually have the designation of AT, followed by a number or group of letters. Each macro is self-contained, having the ability of setting up parameters, performing acquisition, processing the acquired data, possibly setting up new experiments and processing the data acquired from those experiments, creating plots, parameter printouts, archiving the raw data, performing statistical analyses of the data, and writing results to history files and reports.

To better illustrate the structure of these macros, Table 17 gives the source code for macro `AT16`, the turn-on test (channel 1). A column of descriptive comments has been added to clarify the statements.

ATrecord Macro

The `ATrecord` macro is run whenever a numerical result is stored in a history file. It is a general macro that will create the specified history file if it is not present. The macro has a minimum of four arguments: `Name of history file`, `comment line`, `column header line` (name of result) and `value of result`. For example, see the above macro near the bottom. The variables `$turnon` and `$corrcoef` are calculated prior to using the `ATrecord` macro, and these appear in two columns headed by `time` and corr_coef. Note the use of `trunc`. This is necessary to limit the number of decimal places produced.

Up to seven results may be stored in one history file. If this macro is used, be sure to modify the `at_spec_file` in `~/vnmrsys/autotest/atdb` to add the necessary number of lines describing the results and any upper and lower limits desired. If a new history file fails to appear it is usually because of failure to update the `at_spec_file`. A backup copy should be made of the `atdb` in case of accidental overwriting.

The AutoTest macros can be run as independent macros if a specific test is desired. This is done by either entering the macro in the VNMR command line or by selecting the single test from the AutoTest test library panel by using a checkbox and the Begin button. Again, if the file `at_selected_tests` is not empty, the `ATnext` macro will start a new acquisition.

Standard Tests Performed by AutoTest

Automated Console Tests (channel 1 refers to pulsing on channel 1, channel 2 refers to pulsing on channel 2)

- 90° and 30° pulse stability channel 1 and channel 2
- 1 μsec amplitude and turnon stability channel 1 and channel 2
- Pulse turnon time channel 1 and channel 2
- DANTE turnon test channel 1 and channel 2
- Quadrature image: 1 scan and 4 scans
- Quadrature phase selection: 0, 90, 180, and 270 degrees
- Frequency-shifted quadrature image: 1 scan
- Phase stability test (13° test) channel 1 and channel 2
- Phase switch/settling time channel 1 and channel 2
- Attenuator test channel 1 and channel 2 at full and reduced power
- Attenuator test for channel 2 as ^{13}C
- Modulator linearity channel 1 and channel 2
- Small-angle phase shifting 0-360 degrees channel 1 and channel 2
- High-band amplifier compression

Table 17. Source Code for `AT16` Macro Example

`if ($#=0) then`	First time `AT16` is run it has no arguments.
`   ATrtp(’standard’)`	Recalls standard parameter set.
`   text(’Pulse Turnon Test’)`	
`   at_currenttest=turnon_ch1`	Puts name of test in global variable.
`   tpwr=tpwr-6 ph`	
`   array(’pw’,37,0.1,.025)`	Sets up pulse width array.
`   ss=2`	
`   wnt=’ATwft select(celem) aph0 vsadj dssh dtext’`	Specifies what to do every FID
`   wexp=’AT16(‘PART1‘)’`	Specifies what to do at end of experiment.
`   ATcycle`	Disables `wnt` processing if in repeat mode.
`   au`	Begins acquisition and specifies `wnt`/`wexp` processing to occur.
`   write(’line3’,’Pulse Turnon Test (channel 1)’)`	
`   dps`	
`elseif ($1=’PART1’) then`	This part executes at end of experiment.
`   if (at_plotauto=’y’) then`	
`      if (at_printparams=’y’) then`	
`         pap ATpltext`	If parameter printout requested.
`         pps(120,0,wcmax-120,90)`	
`         page`	
`      endif`	
`   endif`	
`   select(arraydim) aph0`	
`   f peak:$ht,cr rl(0) sp=-1p wp=2p vsadj dssh dtext`	
`   ATreg6`	Fits to straight line and displays/plots data.
`   ATpl3:$turnon,$corrcoef`	Determines turn-on time and correlation coefficients
`   $turnon=trunc($turnon) $corrcoef= trunc(1000*$corrcoef)/1000`	Limits number of decimal places.
`   ATrecord(’TURNONch1’,’Pulse Turnon Time (nsec) (channel 1)’,’time  ’,$turnon,’ corr_coef.’,$corrcoef)`	Writes out results to history file.
`   write(’file’,autotestdir+’/REPORT’, ’Pulse Turnon Time (channel 1): %2.0f nsec.-Corr. Coef. = %1.3f ’,$turnon,$corrcoef)`	Writes results to report.
`   if (at_plotauto=’y’) then`	
`      ATpltext(100,wc2max-5)`	
`      full wc=50 pexpl page`	Plots regression fit
`   endif`	
`   ATsvf`	Removes old data set and stores FID under name in at_currenttest.
`   ATnext`	Starts next macro in at_selected_tests file, if present.
`endif`	Closes `elseif` part

- Low-band amplifier compression
- Temperature homogeneity and rise in decoupler heating test
- Temperature increase in spinlock test
- Temperature jump test
- Sensitivity
- AutoGain result for 90° pulse.
- Receiver gain (normal sampling 10-kHz sweep width)
- Receiver gain (oversampling 100-kHz sweep width)
- Folded noise reduction with large spectral width
- Benefit of oversampling at normal gain
- Benefit of oversampling at normal gain + 12dB
- Signal-to-noise as function of gain
- Spectral purity (“glitch test”)
- Lock power test correlation coefficient
- Lock gain test correlation coefficient

Automated Tests with Shaped RF

- Gaussian 90° stability, channel 1 and channel 2
- Gaussian phase stability test, channel 1 and channel 2
- Phase-ramped Gaussian phase stability test, channel 1 and channel 2
- Shaped pulse accuracy- gaussian excitation profile, channel 1 and channel 2
- RF amplitude predictability using a gaussian pulse at variable power, channel 1 and channel 2
- RF amplitude predictability using a gaussian, rectangular and eburp-1 pulses, channel 1 and channel 2
- Amplitude scaling of shaped pulses using a gaussian pulse, channel 1 and channel 2
- RF excitation predictability using a variety of shaped pulses, channel 1 and channel 2

Automated Decoupling Performance Tests

- ^{13}C phase modulation decoupling profiles
  ^{13}C GARP decoupling profile
  ^{13}C WALTZ-16 decoupling profile
  ^{13}C XY-32 decoupling profile
  ^{13}C MLEV-16 decoupling profile
- ^{13}C adiabatic decoupling profiles (if waveform generator present on decoupling channel):
  ^{13}C STUD decoupling profile
  ^{13}C WURST decoupling profile
- Sample heating during ^{13}C broadband decoupling

Automated 90° Pulse Width Calibrations (PW90)

- ^{1}H 90° pulse width calibrations on channels 1 and 2 at high and reduced power
- ^{13}C 90° pulse width calibrations at high and reduced power

RF Homogeneity Tests

- ^{1}H rf homogeneity test
- ^{13}C rf homogeneity test

Gradient Calibrations and Performance Tests

- Gradient level for 10 G/cm along the following:
  Z axis for all gradient probes
  X axis for triax probes
  Y axis for triax probes
  Cancellation following a gradient
- Gradient echo stability for the following:
  Z axis at 30 G/cm
  X axis at 10 G/cm
  Y axis at 10 G/cm
  Z axis at 10 G/cm
- Gradient recovery stability for the following:
  X axis at 10 G/cm
  Y axis at 10 G/cm
  Z axis at 10 G/cm
- Gradient recovery (rectangular and shaped gradients) for the following:
  X axis at ± 10 G/cm
  Y axis at ± 10 G/cm
  Z axis at ± 10 G/cm
- Cancellation after gradient.
  CPMG T_2 result for the following:
- Gradient level = 10 G/cm
  Without gradients
  1% gradient mismatch

Details of AutoTest Experiments

This section provides descriptions of the experiments performed by AutoTest.

All units of the rf system (transmitters, linear modulators, rf attenuators, amplifiers, receivers, and probes) must be in the standard configuration when AutoTest is run. If the system configuration has been changed, it must be returned to the standard configuration before running AutoTest for acceptance testing.

All data is stored, and both plots and statistical analyses are provided as part of the acceptance testing. Plots and statistical analyses are made concurrently with acquisition.

RF Performance Test Descriptions (Nonshaped Channels 1 and 2)

Pulse Stability

Experiment – A single-scan pulse experiment is repeated 20 times and the spectra plotted in a horizontal stack. The average peak amplitude and rms deviation are measured and reported. This test is run for the following:

- 90° flip pulses
- 30° flip pulses
- 1 µsec pulses

Purpose of 90° Pulse Stability – Modern experiments require very high pulse reproducibility to minimize cancellation residuals and T_1 noise in 2D experiments. This test checks amplitude reproducibility by comparing a series of spectra obtained with the signal following a single 90° pulse. The statistical analysis produces an rms deviation, in percent of the average peak height.

Purpose of 30° Pulse Stability – The sinusoidal nature of the excitation profile makes the signal generated following a 90° pulse less sensitive to error than signal following a much smaller flip angle pulse (the top of a sine wave is broad and changes in amplitude less for small changes in flip angle than for a smaller pulse). A 0° flip angle would have the highest sensitivity to flip angle, but would give no signal, of course. A compromise between the extremes of large signal following a 90° pulse, and no signal following a 0° pulse is to use a 30° pulse. The rms deviation is measured from an array of spectra obtained using 30° pulses.

Purpose of 1 µsec Pulse Stability – This test emphasizes the turnon characteristics of the pulse. Any instability of the pulse rise should give a corresponding reduction of measured stability. Since the flip-angle is much less than a 90° or 30° pulse, the measured stability may be lower. The rms deviation is measured from an array of spectra obtained using 1 µsec pulses.

Cancellation Test

Experiment – Four single-scan, 4 two-scan, and 4 four-scan 90° pulse spectra are acquired in which the transmitter phase is held fixed and the receiver is phase-cycled 0, 2, 1, 3. Data are plotted in a horizontal stack with the single-scan spectra on scale. The vertical scale is increased by 100 times and plotted in the same manner. Average residual signal for 2-scan and 4-scan cancellation are determined.

Purpose – Modern experiments (HMQC, HSQC, NOE-difference, etc.) often rely on phase-cycling to achieve desired results. This test compares single-transient response versus two- and four-transient response in which the phase-cycling is set to achieve cancellation.

Phase Stability (13° Phase Error) Test

Experiment – The 90° pulse stability test is repeated but uses a 90° pulse–1 ms—90° pulse train with the carrier positioned 37 Hz off-resonance from the water.

Purpose – Phase stability is essential for high-performance modern experiments. Poor phase stability would produce poorer water suppression and increase T_1 noise in 2D NMR. The most robust tests of phase stability are solids tune-up sequences used for verifying performance for line-narrowing sequences, such as WAHUHA or MREV-8, because these sequences are fairly independent of amplitude stability.

Another test is the 13° test in which two 90° pulses separated by 1 ms are applied with the transmitter placed 37 Hz off-resonance. The resulting NMR response stability is a product of both rf amplitude stability and phase stability because variations in phase between the pulses induce an amplitude change. The observed amplitude error should be divided by a factor of 7.1 to obtain a measure of phase error, in degrees.

Pulse Turn-on Time

- *Experiment* – Single-scan experiments are taken in which the pulse is varied from 0 to 1–2 µs in minimum pulse-width steps at low enough power so that the response is linear. The response is fitted to a straight line and the turn-on time is determined.

  Because of differences in implementation, turn-on times for channel 2 are usually longer than for channel 1, even though the hardware is identical.
- *Purpose* – The quality of modern rf is good enough that examination of pulse shapes using an oscilloscope is not as informative as well-designed and executed NMR tests. The turn-on and turn-off characteristics of a very short pulse are properties that can be measured sensitively by NMR.
- The turn-on test measures the amplitude of a signal following a short variable-length pulse. In the limit of a small flip angle, this dependence is linear. The data are analyzed and least-squares fitted to a straight line. The intercept is the pulse turn-on time.

Attenuator Linearity

Experiment – For a small flip angle pulse, the rf coarse power is varied from maximum to minimum in single-scan mode. The data are plotted in a horizontal stack to facilitate visual inspection. The data are fitted to a linear regression and plotted in phased mode to show any phase change as a function of power.

Purpose – Overall power control is accomplished using PIN diode-controlled rf attenuators. These attenuators are precision devices that should have negligible phase change throughout their full range. The amplitude response should also be logarithmic. A log regression analysis should show the extent of fit to the ideal. The phase change as a function of power is examined. Raw, uncorrected output should be examined without software adjustment of phase and amplitude.

This test does not permit a full assessment of the cause of the phase error, because the amplifier might be in compression at the maximum power output.

Attenuator Linearity at Reduced Power

Experiment – The attenuator linearity test is repeated but with output of the transmitter reduced by the linear modulator. This is done to isolate the effect of the rf amplifier.

Purpose – The attenuator linearity test is performed, but with reduced power input to the attenuator (using the linear modulator to reduce the output power from the transmitter). Raw, uncorrected output should be examined without software adjustment of phase and amplitude corrections.

Linear Modulator Linearity Tests

Experiment – With the coarse rf amplitude set at a value 23 dB down from maximum, the rf power is varied using the linear modulator. The linear modulator is used for fine power control and shaped rf excitation. The rf amplitude is varied, over a range of 60 dB, in 100 equally-spaced steps over the whole range. This should produce a linear ramp of signal response following a small flip-angle pulse. Spectra are plotted in a horizontal stack of spectra in phased mode with the highest signal full scale. The width of the plotted region around the water is set narrow enough to clearly show the base of the water peak. The data are fitted to a straight line using a linear regression analysis and plotted.

Purpose – Further power control is possible using the linear modulator present on the NMR transmitter board. This test produces a series of experiments in which the pulse power is changed over the full range of the modulator. The linear nature is tested by a linear least-squares fit of the data.

Predictable power control is essential for delivering accurate shaped pulses and for precise power level control in Hartmann-Hahn experiments in both solids and liquids. Raw, uncorrected output should be examined without software adjustment of phase or amplitude.

Linear Modulator Linearity Tests with Attenuators Set to Full Attenuation

Experiment – The linear modulator linearity test is repeated, with the coarse rf amplitude control set for minimum power (resulting in a maximum attenuation of 139 dB). The pulse width is increased correspondingly to obtain comparable signal-to-noise as in the linear modulator linearity test. The data are fitted to a straight line using a linear regression analysis and plotted.

Pulse Shape Test—DANTE

Experiment – The rf amplitude is set for a 20 µs 90° pulse, and the result is compared to that for a single-scan spectrum:

- 10 pulses, 2 µs each
- 20 pulses, 1 µs each
- 25 pulses, 800 µs each
- 50 pulses, 400 µs each
- 100 pulses, 200 µs each

For all except the first 20 µs pulse, a 1 µs delay is inserted between each pulse. The data are plotted in a horizontal stack to permit comparison of amplitudes. The amplitudes are measured and printed.

Purpose – A DANTE-type test is performed in which the signal response following a 20-µs pulse is measured. This is compared with a series of experiments in which the pulse is increasingly divided into series of pulses spaced by 1 µs. The sum of the pulses is held constant at 20 µs.

If the pulse shape is *ideal* and the total time of the pulse train is short compared to T_2, the rotation of magnetization should be identical. As the pulse length shortens, any non-ideality of pulse shape is revealed as a reduction in intensity.

Phase Switch Settling Time

Experiment – Parameters `p1` and `pw` are set to the same value (1 µs) and 30 spectra are acquired using a 2-pulse sequence, with the first and second pulses separated by a delay of 20 µs. The phase of the second pulse is shifted 180° from the first pulse at a variable time prior to the second pulse. A single-pulse spectrum is also acquired. The arrayed 2-pulse spectra and the single-pulse spectrum are plotted, with the single-pulse spectrum last. This last spectrum serves as a reference. The phase shift should be accomplished in 100 ns or less.

Purpose – This test exercises the phase-shift hardware by finding the time needed to perform a 180° phase shift. The pulse sequence is a version of *jump-and-return* in which two 1-µs pulses are executed just 20 µs apart. Ideally, because the second pulse has a 180° phase shift with respect to the first pulse, there should be no excitation. By varying the time before the second pulse, at which the phase shift is done, from 0 to 20 µs, an estimate of the phase switch and settling time can be made. The last spectrum is that from just a single, 1-µs pulse, and serves as a reference.This phase shift should be accomplished in 100 ns or less.

RF Homogeneity

^{1}H RF Homogeneity Experiment – One hundred experiments are run in which the pulse width is incremented from 1 to 100 µs. The spectra are plotted in a horizontal stack in phased mode and sufficiently expanded so that the base of the water can be examined using the same phase settings for each spectrum (use channel 1).

^{13}C RF Homogeneity Experiment – In the pulse sequence

`delay—pw90(1H)—delay(1/2JCH)—pw(13C)`

the `pw(`^{13}`C)` is varied from 0 to a flip angle greater than 900 ° while observing the ^{13}C-coupled protons. The 0-flip-angle spectrum is adjusted to full scale and the data expanded to show only the ^{13}C-bound protons side-by-side to permit measurement of X-coil rf homogeneity. The results are plotted and displayed in magnitude mode showing one of the lines of the methanol doublet.

Purpose – This test checks the homogeneity of the rf field strength throughout the active region of the sample. In an ideal case, for nuclei having reasonable T_2 values, the signal generated following a 360° + 0° pulse should match that following a 0° pulse. The signal strength as a function of flip angle should be sinusoidal. The amount of drop off is related to the inhomogeneity of the rf field (RF homogeneity results are shimming-dependent. If results for a given probe are outside of specifications, reshim the sample or use the sample specified in the probe test manual).

High rf homogeneity is important because many important pulse sequences use a large number of pulses. The signal losses accumulate with each pulse such that, in worst cases, all the desired signal is lost. Most heteronuclear, indirect detection experiments on large molecules use HSQC pulse sequence components. These contain 6 to 10 ^{1}H pulses, including 4 to 8 X nucleus 180° pulses. High rf homogeneity is especially important in these cases.

Receiver Test

Experiment – Single scan spectra are collected that span the range of receiver gain and divide that range into at least 25 evenly spaced values of gain, including the highest and lowest gain values.The data are plotted with the highest signal on scale so that the heights can be easily compared.

The results are fitted to a straight line using linear regression, and the fitted data are plotted. Next, the data are normalized and plotted with the water signal held to a constant height so that the noise levels are easily compared (a few mm of noise in the baseline are provided).

The signal-to-noise ratios for the water line in all spectra are measured with a spectral width of 10000 Hz and no oversampling. Channel 1 is used for the acquisition. With oversampling ×10, the experiment is repeated. Processing, plotting, and quantization of the oversampled data are the same as for the data from the 10000-Hz experiment.

Purpose – Receiver gain is selectable in a logarithmic manner (in dB). In an ideal case, variation of receiver gain should produce a logarithmic dependence of signal strength. As the gain is lowered, the noise becomes dominated by noise generated in the ADCs, not in the preamplifier and probe. Regardless of the signal strength, operation in this range of gain will produce poorer signal-to-noise.

Image Rejection Test

Experiment – Plot the data from the following tests first in a horizontal stack, with the single-scan data on-scale, and then with the vertical scale increased 100 times. Quantitate the average image and center glitch.

- Four single-scan and four 4-scan 90° pulse spectra are acquired in which the carrier frequency is shifted 1000 Hz from the water. The carrier position is not changed during the pulse sequence and acquisition, and digital filtering is not used.
- The test is repeated using FSQD with single scans.

Purpose – This test checks the inherent balance in the two quadrature channels and the ability of phase cycling to eliminate any quadrature image. Four single-transient and 4 four-transient responses are collected and compared.

Quadrature images can also be eliminated using digital filtering techniques. The FSQD test measures image rejection under these conditions.

Quadrature Phase Selection

Experiment – Plot the data from the following tests first in a horizontal stack, with the single-scan data on- scale. Quantitate the average intensities.

- With constant receiver phase, acquire multiple spectra with the observe pulse set to a phase of 0, 90, 180 and 270 degrees. Phase the 0 degree spectra to positive absorption. The spectra should show absorption, dispersion(+), absorption(-) and dispersion(-). Calculate the average and standard deviation of the spectra grouped by phase.

Purpose – This test checks the quadrature phase shift which is used within any phase-cycling experiments

Shaped Pulse Test Descriptions (Channels 1 and 2)

Gaussian-Shaped Pulse Excitation

Experiment – A gaussian-shaped pulse, with excitation bandwidth at 50% amplitude about 200 Hz, is applied (e.g., a 12-ms, 90° pulse length). Single-scan spectra are taken with the transmitter stepped over the range ±250 Hz from resonance, in 5–Hz steps.

The data are plotted in a horizontal stack, with the on-resonance spectrum at full scale to illustrate the gaussian shape of excitation. The vertical scale is increased by ×10 and plotted again to show the wings.

Purpose – The most demanding test of shaped pulse accuracy is the ideality of the NMR data following a shaped pulse. This test determines the accuracy of a gaussian pulse by examination of the off-resonance excitation. This is done by repeating the same single-pulse excitation while varying the transmitter position through a wide range.

A stacked array of data should show the magnitude of excitation as a function of offset from resonance. In the ideal case, this excitation envelope would also be gaussian. Any non-gaussian nature of the pulse, *as delivered to the probe*, would be represented by a convolution of excitation envelopes. For example, if the power were not delivered in a linear manner, producing some rectangular nature, the excitation envelope would have some sin*x*/*x* nature, producing characteristic sinc wiggles. The lack of such non-gaussian behavior is a direct measure of the accuracy with which the hardware can deliver an ideal shape to the nuclei.

Gaussian 90° Pulse Stability

Experiment – The rf 90° pulse stability test is repeated using a gaussian pulse at the same peak power.

The data are plotted in a horizontal stack, with the on-resonance spectrum at full scale to illustrate the gaussian shape of excitation.

Purpose – Modern experiments require very high pulse reproducibility to minimize cancellation residuals and T_1 noise in 2D experiments. This tests amplitude reproducibility by comparing a series of spectra obtained with the signal following a single 90° pulse. The statistical analysis produces an rms deviation, in percent of the average peak height.

Gaussian 13° Phase Error

Experiment – The rf 13° test is repeated using a gaussian pulse at the same peak power as in the phase stability test.

The data are plotted in a horizontal stack, with the on-resonance spectrum at full scale.

Purpose – The rf 13° test can be done using shaped pulses. In this case, a gaussian pulse is used at high peak power.

Gaussian SLP 13° Phase Error (Phase-Ramped Gaussian Pulses)

Experiment – This 13° test is repeated using a phase-ramped gaussian pulses. The rf carrier should be 37 Hz off-resonance from water, but the center of excitation of the gaussian phase-ramped pulses should be 1000 Hz from the carrier. The amplitude of the gaussian pulses is set high enough to exert a 90° pulse on the water.

Purpose – The 13° test can be done using phase-modulated pulses. These types of pulses provide single- or multiple-frequency selective excitation through the use of both amplitude and phase modulation.

Shaped Pulse Settability

Experiment – Single-pulse, single-scan spectra are collected. The rf power is dropped in eight successive spectra by 3 dB each time and the pulse width increased so that a 90° flip angle is maintained. The spectra are plotted in a horizontal stack for easy amplitude comparison.

Purpose – An rf attenuator should permit accurate power control. In this case, the pulse length is repeatedly incremented while appropriately reducing power levels. The NMR response should be identical.

Shaped Pulse Test – Rectangular, Gaussian, and EBURP-1

Experiment – Single-scan, one-pulse excitation spectra are collected using rectangular, gaussian, and EBURP-1 pulses at the same peak amplitude (note the power value and pulse lengths). Constant peak amplitude is maintained; therefore, pulse width ratios of 1.0:2.4:16.0 for the rectangular:gaussian:EBURP-1 pulses, respectively, are used to obtain the same flip angle. Spectra are plotted side-by-side in absolute intensity mode at full vertical scale.

At any constant power, the 90° pulse lengths should reflect their theoretical ratios. Here, the pulse lengths are set in a ratio of 1:2.4:16. The resulting NMR responses should be identical in amplitude.

Shaped Pulse Test—Constant Bandwidth for a Variety of Shapes

Experiment – Single-scan, one-pulse excitation spectra are collected using a variety of shapes that are automatically calculated using Pbox, based on a single pulse calibration using a rectangular pulse, for a constant 4000 Hz bandwidth. The shaped pulses have different peak amplitudes and pulse widths (note the power value and pulse lengths). Spectra are plotted side-by-side in absolute intensity mode at full vertical scale. The resulting NMR responses should be identical in amplitude.

Shaped Pulse Scalability

Experiment – A small flip-angle gaussian pulse is used for a single-transient, one-pulse spectrum. The linear modulator is used to scale down the amplitude of the pulse in 100 steps over a range of 60 dB. Plot widths are set small enough to show the base of the water and plot all spectra in a horizontal stack in phased mode with the maximum signal spectrum at full scale.

Purpose – The linear nature of the system is graphically tested by measuring NMR response when the amplitude is under full control, both by the rf attenuator and by the linear modulator.

^{13}C Test Descriptions

X-coil rf homogeneity can be determined using an indirect detection pulse sequence. Sensitivity in many indirect detection experiments is markedly affected by X-coil performance because of the large number of 180° pulses used.

X-decoupling is tested for various modulation schemes at constant amplitude (WALTZ-16, GARP-1, etc.) as well as more powerful adiabatic pulse techniques. Efficiency is measured by varying the ^{13}C decoupling frequency while observing the proton spectrum under broadband decoupling.

^{13}C 90° Pulse Width Calibration

The power level for a 90° flip of approximately 15 µs on the X-coil of the probe is determined. Amplifier compression is determined by lowering power by 12 dB and redetermining the 90° pulse width. Both results are reported.

X-Phase Modulation Decoupling Profiles

^{13}C power level is reduced 20 dB from the level used to obtain a 15 µs 90° (approximately 1.8 kHz), and the ^{13}C decoupling efficiency is determined for the following phase-modulated, constant-amplitude broadband decoupling sequences:

- WALTZ-16
- GARP-1
- XY-32
- MLEV-16

The ^{13}C decoupling frequency is varied over a range of ± 80 ppm in a series of single-scan, proton-observe experiments. Only the ^{13}C-bound protons are shown in the expanded spectrum, which is plotted with spectra side-by-side in absolute intensity mode to illustrate decoupling efficiency.

X-Adiabatic Decoupling Profiles

The decoupling profile experiment is repeated with the following adiabatic decoupling schemes:

- STUD modulation
- WURST modulation

Decoupler Heating

The same test as in the variable temperature test is performed but this time using a 75-ms ^{13}C decoupling period prior to acquisition within a total recycle time of 1.5 seconds, including acquisition. One-hundred, single-scan spectra are collected with ^{13}C decoupling followed by 100 identical spectra with no decoupling. The spectra are plotted in a stacked manner to permit examination of the rate of change of temperature, the homogeneity of temperature, and the length of time necessary to reach equilibrium. The rf field strength must be sufficient to decouple over a 160 ppm range using garp-1.

Gradient Tests Descriptions

Gradient Profile

Experiment – A spectrum with a 100 kHz spectral width is acquired using a gradient echo (collect echo during a Z-axis gradient). This acquisition is repeated for both positive and negative gradients that are sufficient to spread the pattern greater than 50 kHz at the base. Gradient strength and duration as well as the size of the active length of the coil are noted. The experiment is repeated for both the X-axis and Y-axis gradients if available.

Purpose – This test uses pulsed field gradients (PFGs) to quantitate the gradient field strength. The width of the pattern is directly proportional to the gradient strength. The width at 20% of maximum is used to calculate the gradient strength. Both positive and negative gradients are used. This should be done for all orthogonal axes available.

Field Recovery Stability

Experiment – The 90° pulse stability test is performed but, preceding the rf pulse, is a 1-ms, 30-G/cm Z-axis gradient pulse, which is then followed by a 100–µs field stabilization delay. This test is repeated with a 10 G/cm gradient pulse. If X and Y gradients are available, the test is repeated with a 10 G/cm gradient pulse for each gradient.

Purpose – A gradient is applied prior to a measuring pulse. The stability of the signal response is used to measure the ability and reproducibility of the system to recover from the gradient pulse.

Field Recovery

Experiment – The gradient 90° pulse stability test is repeated with the field recovery delay varied from 0 to 1000 µs, using a positive 10 G/cm Z-axis gradient pulse. The spectra are plotted in a horizontal stack with the 1000 µs data at full scale. The test is repeated using a negative 10 G/cm gradient and, if available, for both the X and Y gradients. Rectangular and half sine gradients are used. Recovery is defined as the time it takes to recover to 95% or more of the amplitude at 1000 us.

Purpose – A gradient is applied prior to a measuring pulse and the time before the pulse is varied. The rate of recovery determines how soon a pulse may be applied.

Gradient Echo Stability

Experiment – The 90° pulse stability test is run, this time with a positive gradient for 1 ms, a 500 µs delay, and a negative gradient for 1 ms following the rf pulse. The following gradient strengths and axes are used.

- 30 G/cm—Z axis only
- 10 G/cm—Z axis and, if present, Y and Z axes

Purpose – Following a single pulse, a pair of oppositely-signed gradients is applied. The stability of the resulting refocused signal measures any instability in the gradient amplitude, as well as the accuracy of the gradient level control. This should be done for all orthogonal axes available.

Gradient Effect on Cancellation Test

Experiment – Four 1-scan, four 2-scan and four 4-scan 90° pulse spectra are acquired, with a 10 G/cm Z-axis gradient pulse 100 µs prior to the rf pulse and the transmitter phase held constant while the receiver is phase-cycled 0, 2, 1, 3. The spectra are plotted in a horizontal stack with the single-scan spectra on scale. Vertical scale is increased by 100 times and plotted in the same manner. The average residual signal for the 4-scan cancellation is quantitated and the results plotted and analyzed as above.

Purpose – The cancellation test is done with a gradient pulse applied 100 µs before the rf pulse. If the lock circuitry and field recovery characteristics are favorable, no deterioration in cancellation efficiency should be noted.

CPMG T_2

Experiment – A CPMG T_2 experiment is preformed with 2 ms between 180° pulses for total echo pulse trains ranging from a few milliseconds out to at least 2*T_1. This experiment is repeated for the case where 500 µs 10 G/cm rectangular pulses are placed around the 180° pulses in each echo, as well as the case in which no gradients are used. The values for T_2 are reported for all cases. The experiment is repeated for the case of a 1% mismatch in gradient amplitude.

Purpose – Following a single pulse, pairs of same-signed gradients are applied within the echoes of a CPMG T_2 pulse echo train. The measured T_2 gives a measure of any instability in the gradient amplitudes as well as the accuracy of the gradient level control. This measurement is compared to an identical experiment in which the gradient amplitudes are set to zero or mismatched by 1%. More rapid diffusion at higher temperature will cause the gradient/no gradient comparison to worsen. Therefore, performance over time should be compared for the same temperature. (Measurements done at ~4° C have shown no difference in measured T2 for the gradient/no gradient cases, indicating that diffusion is responsible for the difference in these cases at higher temperatures).

^{13}C Power-Limited Pulse Tests

^{13}C pw90 (15 µsec) and Lowband Amplifier Compression

Experiment: The user-limited attenuator setting for a 90° flip of approximately 15 µs on the X-coil of the probe is determined. This setting is limited to an upper limit of the user-supplied value. At this power level the pulse width is varied to obtain a pw90 value. Amplifier compression is determined by lowering power by 12 dB and redetermining the 90° pulse width. The results are reported and stored in a separate history file from the standard history file (C^{13}PW90c instead of C^{13}PW90).

^{13}C pw90 (10 µsec) and Lowband Amplifier Compression

Experiment: The user-limited attenuator setting for a 90° flip of 10 µs on the X-coil of the probe is determined. This setting is limited to an upper limit of the user-supplied value. Amplifier compression is determined by lowering power by 12 dB and redetermining the 90° pulse width. The results are reported and stored in a separate history file from the standard history file (C^{13}PW90c_10 µs instead of C^{13}PW90). This test is only used on systems having a low band amplifier and probe capable of this rf field strength (typically 800 MHz systems)

^{13}C Power for pw90=15.0 µsec and Lowband Amplifier Compression

Experiment: The optimum user-limited attenuator level for a 90° flip of 15.0 µs on the X-coil of the probe is determined. The power level is limited to an upper limit of the user-supplied value. At this attenuator level the fine power modulator is varied to obtain a 90°

pulse. Amplifier compression is determined by lowering power by 12 dB and redetermining the 90° pulse width. The results are reported and stored in a separate history file from the standard history file (C^{13}PW90fc_15 µs instead of C^{13}PW90).

^{13}C Power for pw90=10.0 µsec and Lowband Amplifier Compression

Experiment: The optimum user-limited attenuator level for a 90° flip of 10.0 µs on the X-coil of the probe is determined. The power level is limited to an upper limit of the user-supplied value. At this attenuator level the fine power modulator is varied to obtain a 90° pulse. Amplifier compression is determined by lowering power by 12 dB and redetermining the 90° pulse width. The results are reported and stored in a separate history file from the standard history file (C^{13}PW90fc_10usec instead of C^{13}PW90). This test is only used on systems having a low band amplifier and probe capable of this rf field strength (typically 800 MHz systems)

^{13}C RF Homogeneity at Limited Power (15 µs)

Experiment: The power level is limited to an upper limit of the user-supplied value. A power sufficient to produce a ~15 µs pw90 is used and the ^{13}C pulse width is incremented. The data are plotted in a horizontal stacked array to allow inspection for arcing or phase instability. Based on the relevant maxima, a new array of 0, 360 and 720 degree ^{13}C pulses is used to more accurately determine the rf homogeneity. The results are reported and stored in a separate history file from the standard history file (C^{13}RFHOMOc).

^{13}C RF Homogeneity at Limited Power(10 µs)

Experiment: The power level is limited to an upper limit of the user-supplied value. A power sufficient to produce a ~10 µs pw90 is used and the ^{13}C pulse width is incremented. The data are plotted in a horizontal stacked array to allow inspection for arcing or phase instability. Based on the relevant maxima, a new array of 0, 360 and 720 degree ^{13}C pulses is used to more accurately determine the rf homogeneity. The results are reported and stored in a separate history file from the standard history file (C^{13}RFHOMOc_10µsec). This test is only used on systems having a low band amplifier and probe capable of this rf field strength (typically 800 MHz systems)

^{13}C RF Homogeneity at Limited Power (pw90=15.0 µs)

Experiment: The power level is limited to an upper limit of the user-supplied value. The attenuator and modulator powers found for pw90=15.0usec are used and the ^{13}C pulse width is incremented. The data are plotted in a horizontal stacked array to allow inspection for arcing or phase instability. Based on the relevant maxima, a new array of 0, 360 and 720 degree ^{13}C pulses is used to more accurately determine the rf homogeneity. The results are reported and stored in a separate history file from the standard history file (C^{13}RFHOMOfc_15µsec).^{13}C RF Homogeneity at Limited Power (pw90=10.0 µs)

^{13}C RF Homogeneity at Limited Power (pw90=10.0 µs)

Experiment: The power level is limited to an upper limit of the user-supplied value. The attenuator and modulator powers found for pw90=10.0 µs are used and the ^{13}C pulse width is incremented. The data are plotted in a horizontal stacked array to allow inspection for arcing or phase instability. Based on the relevant maxima, a new array of 0, 360 and 720 degree ^{13}C pulses is used to more accurately determine the rf homogeneity. The results are reported and stored in a separate history file from the standard history file (C^{13}RFHOMOfc_10usec). This test is only used on systems having a low band amplifier and probe capable of this rf field strength (typically 800 MHz systems)

^{13}C RF Amplifier Linearity at Limited Power(pw90=15 µs)

Experiment – The ^{13}C pw360 is determined indirectly as a function of attenuator setting for ^{13}C on channel 2. The data are fitted to a linear regression. The maximum power is the attenuator setting found to produce a 15 µs pw90.

Purpose The amplitude response should be logarithmic. A log regression analysis should show the extent of fit to the ideal. The results are reported and stored in the history C^{13}AMPc.

^{13}C RF Amplifier Linearity at Limited Power(pw90=10usec)

Experiment – The ^{13}C pw360 is determined indirectly as a function of attenuator setting for ^{13}C on channel 2. The data are fitted to a linear regression. The maximum power is the attenuator setting found to produce a 10 µs pw90.

Purpose The amplitude response should be logarithmic. A log regression analysis should show the extent of fit to the ideal. The results are reported and stored in the history C^{13}AMPc_10usec.

^{13}C/^{15}N Power-Limited Decoupling Tests

^{13}C/^{15}N Decoupling Noise (FID) at User-Selected Decoupling Power

Experiment: The time-domain noise characteristics in a single-scan experiment with no observe pulse is measured with: (1) no decoupling; (2) ^{13}C decoupling,; (3) ^{15}N decoupling; and (4) ^{13}C/^{15}N decoupling, with power levels for decoupling set at the user-supplied limits. The combined ^{13}C/^{15}N decoupling has 3 dB reduction for each channel from the user-supplied maxima. ^{15}N decoupling tests are skipped if the user-supplied maximum ^{15}N pulse power is set to zero. Average RMS noise, real/imaginary RMS, and real/imaginary dc offset results are stored in separate history files (NOISEc, NOISECc, NOISENc, and NOISECNc). Real and imaginary fids are plotted. Spectra are plotted.

Purpose Decoupling should add minimal noise. This tests permits a direct measurement of time-domain noise arising from decoupling at maximum permitted decoupling powers.

^{13}C/^{15}N Decoupling Noise (FID) as Function of Decoupling Power

Experiment: The time-domain average RMS noise in a single-scan experiment with no observe pulse is measured with: (1) no decoupling; (2) ^{13}C decoupling,; (3) ^{15}N decoupling; and (4) ^{13}C/^{15}N decoupling, with power levels for decoupling varied from 1 dB up to the user-supplied limits. The combined ^{13}C/^{15}N decoupling power has a 3 dB reduction for each channel from the user-supplied maxima. ^{15}N decoupling tests are skipped if the user-supplied maximum ^{15}N pulse power is set to zero. Average RMS noise is measured at each power level and the average of all these is stored along with the corresponding effective “loss” of signal-to-noise ratio (expressed in % loss). The average RMS noise level and corresponding power levels are stored in a text file as the experiment proceeds. At the end, a histogram of the results is displayed and (optionally) plotted. The results are stored in separate history files (NOISE_c, NOISE_C_c, NOISE_N_c, and NOISE_CN_c)

Purpose This tests permits a direct measurement of time-domain noise arising from decoupling as a function of power.

Sensitivity as Function of ^{13}C Decoupling Power

Experiment: A single-scan experiment with a 90 degree observe pulse is measured with no decoupling and ^{13}C decoupling for decoupling power levels varied from 1 dB up to the user-supplied limit.

Signal-to-noise of the water is measured at each power level and the average of all these is stored along with the corresponding effective “loss” of signal-to-noise ratio relative to the no-decoupling results (expressed in % loss). The average signal-to-noise level and corresponding power levels are stored in a text file as the experiment proceeds. At the end, a histogram of the results is displayed and (optionally) plotted. A fixed region of noise is plotted for each power level to permit visual comparison. Plotted data also has printouts of signal-to-noise measurements and line widths for each spectrum.

The test is performed separately for cw, hardware modulator-based waltz-16 and waveform generator-based waltz-16 modulation.Results are stored in the SN_^{13}Cdec_*_c history files.

Purpose This tests permits a direct measurement of frequency-domain noise arising from decoupling as a function of power, for different modulation schemes involving different hardware.

Sensitivity as Function of ^{13}C Decoupling Power in Presence of ^{15}N Decoupling

Experiment: A single-scan experiment with a 90° observe pulse is measured with no decoupling; and ^{13}C decoupling with power levels for decoupling varied from 1 dB up to the user-supplied limit less 6 dB. ^{15}N decoupling at the user-supplied limit less 3 dB is applied in all experiments.

Signal-to-noise of the water is measured at each power level and the average of all these is stored along with the corresponding effective “loss” of signal-to-noise ratio relative to the no-decoupling results (expressed in % loss). The average signal-to-noise level and corresponding power levels are stored in a text file as the experiment proceeds. At the end, a histogram of the results is displayed and (optionally) plotted

A fixed region of noise is plotted for each power level to permit visual comparison. Plotted data also has printouts of signal-to-noise measurements and line widths for each spectrum.

The test is performed separately for cw, hardware modulator-based waltz-16 and waveform generator-based waltz-16 modulation. Results are stored in the SN_^{15}Ncw_13Cdec_*_c history files.

Purpose This tests permits a direct measurement of noise arising from simultaneous ^{13}C and ^{15}N decoupling as a function of power, for different modulation schemes.

Sensitivity as Function of ^{15}N Decoupling Power

Experiment: A single-scan experiment with a 90° observe pulse is measured with no decoupling and ^{15}N decoupling for decoupling power levels varied from 1 dB up to the user-supplied limit.

Signal-to-noise of the water is measured at each power level and the average of all these is stored along with the corresponding effective “loss” of signal-to-noise ratio relative to the no-decoupling results (expressed in % loss). The average signal-to-noise level and corresponding power levels are stored in a text file as the experiment proceeds. At the end, a histogram of the results is displayed and (optionally) plotted. A fixed region of noise is plotted for each power level to permit visual comparison. Plotted data also has printouts of signal-to-noise measurements and line widths for each spectrum.

The test is performed separately for cw, hardware modulator-based waltz-16 and waveform generator-based waltz-16 modulation. Results are stored in the SN_^{15}Ndec_*_c history files.

Purpose: This tests permits a direct measurement of noise arising from decoupling as a function of power, for different modulation schemes involving different hardware.

Sensitivity as Function of ^{15}N Decoupling Power in Presence of ^{13}C Decoupling

Experiment: A single-scan experiment with a 90° observe pulse is measured with no decoupling; and ^{15}N decoupling with power levels for decoupling varied from 1 dB up to the user-supplied limit less 6 dB. ^{13}C decoupling at the user-supplied limit less 3 dB is applied in all experiments.

Signal-to-noise of the water is measured at each power level and the average of all these is stored along with the corresponding effective “loss” of signal-to-noise ratio relative to the no-decoupling results (expressed in % loss). The average signal-to-noise level and corresponding power levels are stored in a text file as the experiment proceeds. At the end, a histogram of the results is displayed and (optionally) plotted

A fixed region of noise is plotted for each power level to permit visual comparison. Plotted data also has printouts of signal-to-noise measurements and line widths for each spectrum.

The test is performed separately for cw, hardware modulator-based waltz-16 and waveform generator-based waltz-16 modulation. Results are stored in the SN_^{13}Ccw_^{15}Ndec_*_c history files.

Purpose: This tests permits a direct measurement of noise arising from simultaneous ^{13}C and ^{15}N decoupling as a function of power, for different modulation schemes.

^{13}C Decoupling using Phase Modulation

^{13}C power level is reduced 20 dB from the hard-pulse level, limited by the user-supplied upper limit, and the ^{13}C decoupling efficiency is determined for the following phase-modulated, constant-amplitude broadband decoupling sequences using the built-in channel 2 hardware modulator (not the waveform generator):

The ^{13}C decoupling frequency is varied over a range of ± 80 ppm in a series of single-scan, proton-observe experiments. Only the ^{13}C-bound protons are shown in the expanded spectrum, which is plotted with spectra side-by-side in absolute intensity mode to illustrate decoupling efficiency.

Purpose This tests the hardware modulator under the user-supplied upper limit for decoupling.

^{13}C Decoupling using Adiabatic Decoupling

The decoupling profile experiment is repeated using the (optional) channel 2 waveform generator with two adiabatic decoupling schemes (STUD and WURST) under user-supplied upper limit on power. The waveforms are created automatically based on the calibrations produced in the AutoTest run and the user-supplied upper limit on power.

Purpose: This tests the waveform generator under the user-supplied upper limit for decoupling.

Decoupler Heating

The same test as described above is performed, subject to the user-supplied upper limit on decoupling power. Results are stored in the history file `DECHEATc`.

Amplitude Stability in the Presence of ^{13}C Decoupling

The 90° stability test is performed in the presence of ^{13}C decoupling at the user-supplied maximum decoupling power.

Purpose: Decoupling should not degrade amplitude stability which could lead to degradation of indirect detection experiments.

Amplitude Stability in the Presence of ^{15}N Decoupling

The 90° stability test is performed in the presence of ^{15}N decoupling at the user-supplied maximum decoupling power. If the user-supplied maximum ^{15}N pulse power is set to zero, this test is skipped.

Purpose: Decoupling should not degrade amplitude stability which could lead to degradation of indirect detection experiments.

Amplitude Stability in the Presence of Combined ^{15}N and ^{13}C Decoupling

The 90° stability test is performed in the presence of combined ^{15}N and ^{13}C decoupling at the user-supplied maximum decoupling powers less 3 dB each.

Purpose: Decoupling should not degrade amplitude stability which could lead to degradation of indirect detection experiments.

Decoupled Methanol Amplitude Stability using a 6 kHz ^{13}C Decoupling rf Field

The ^{13}C-decoupled methanol signal is recorded under the user-supplied upper limit for ^{13}C decoupling power. The ^{13}C decoupling power is set to achieve a 6 kHz rf amplitude for 100 msec and a 1 second relaxation delay using waltz-16 modulation. The experiment is done both on- and off-resonance in ^{13}C frequency. The off-resonance frequency is determined automatically to be at the edge of the waltz-16 effective bandwidth. Amplitude stability is measured for both and stored in a history file.

Purpose: It is important to characterize both on- and off-resonance decoupling since most experiments are designed for a wide variety of ^{13}C chemical shifts. Incomplete or unstable decoupling would be most damaging at the extremes of decoupling bandwidths. This test measures this effect.

Decoupled Methanol Amplitude Stability using a 3kHz ^{13}C Decoupling RF Field

The ^{13}C-decoupled methanol signal is recorded under the user-supplied upper limit for ^{13}C decoupling power. The ^{13}C decoupling power is set to achieve a 3kHz rf amplitude for 100 msec and a 1 second relaxation delay using waltz-16 modulation. The experiment is done both on- and off-resonance in ^{13}C frequency. The off-resonance frequency is determined automatically to be at the edge of the waltz-16 effective bandwidth. Amplitude stability is measured for both and stored in a history file.

Purpose: It is important to characterize both on- and off-resonance decoupling since most experiments are designed for a wide variety of ^{13}C chemical shifts. Incomplete or unstable decoupling would be most damaging at the extremes of decoupling bandwidths. This test measures this effect.

Comparison of Single-Pulse vs. HSQC Decoupled Methanol

Experiment: Single-scan experiments with a 90° observe pulse are measured with ^{13}C decoupling. The same acquisition conditions are used except that a phase-cycled ^{13}C HSQC pulse sequence is used. The intensities and signal-to-noise ratios for each are averaged and standard deviations calculated. The ratio of the average peak intensities and ratio of the average signal-to-noise ratios are calculated. The results are stored in the `C13SNc` history file.

Purpose: This test permits a direct measurement of the impact which the probe's rf homogeneity and the use of a multi-pulse sequence has in determining effective s/n in indirect detection experiments. In most probes the ^{1}H observe coil will detect signal from parts of the sample above and below the coil “window”. This signal contributes to s/n in a single-pulse experiment, but does not in an X-edited experiment. The HSQC peak height is determined by, first, the portion of the sample accessible by the X-rf field (nominally the “window” size) and, second, by the X-coil rf homogeneity itself.

Comparison of Single-Pulse vs. HSQC Decoupled Methanol (^{13}C pw90=10 µsec)

Experiment: Single-scan experiments with a 90° observe pulse are measured with ^{13}C decoupling. The same acquisition conditions are used except that a phase-cycled ^{13}C HSQC pulse sequence is used. The intensities and signal-to-noise ratios for each are averaged and standard deviations calculated. The ratio of the average peak intensities and ratio of the average signal-to-noise ratios are calculated. The results are stored in the C^{13}SNc_10µsec history file.

Purpose: This test permits a direct measurement of the impact which the probe’s rf homogeneity and the use of a multi-pulse sequence has in determining effective s/n in indirect detection experiments. In most probes the ^{1}H observe coil will detect signal from parts of the sample above and below the coil “window”. This signal contributes to s/n in a single-pulse experiment, but does not in an X-edited experiment. The HSQC peak height is determined by, first, the portion of the sample accessible by the X-rf field (nominally the “window” size) and, second, by the X-coil rf homogeneity itself.

^{15}N Power-Limited Pulse and Decoupling Tests

If the standard AutoTest sample is augmented with ~0.1% acetonitrile(^{15}N, 99%), several tests analogous to those above are available for testing ^{15}N performance (using an indirect or triple-resonance probe). The tests are available for either channel 2 or channel 3 (with appropriate cabling). Results are stored in appropriate history files.

^{15}N `pw90` and Lowband Amplifier Compression

Experiment: The user-limited attenuator is used. At this power level the pulse width is varied to obtain a `pw90` value. Amplifier compression is determined by lowering power by 12 dB and redetermining the 90° pulse width.

^{15}N RF Amplifier Linearity at Limited Power

Experiment: The ^{15}N `pw90` is determined indirectly as a function of attenuator setting for ^{15}N.

The data are fitted to a linear regression.

Purpose: The amplitude response should be logarithmic. A log regression analysis should show the extent of fit to the ideal.

Acetonitrile ^{15}N-HMQC Amplitude Stability

Experiment: The ^{15}N HMQC acetonitrile signal reproducibility is recorded in multiple trials under the user-supplied upper limit for ^{15}N pulse power.

Purpose: ^{15}N indirect detection needs stable ^{15}N pulses to minimize `t1` noise in 2D/3D experiments. This test measures the stability of the ^{15}N-edited spectrum.

Acetonitrile ^{15}N-HMQC Decoupler Heating

Experiment: The ^{15}N HMQC acetonitrile signal is recorded under the user-supplied upper limit for ^{15}N pulse power with 75 msec of decoupling at 2 KHz (scaled down for lower field systems) or the user-supplied upper limit for ^{15}N decoupling power. The temperature rise is measured. Plotted data show the speed to equilibrium and the temperature homogeneity during the process.

Purpose: ^{15}N indirect detection needs stable temperatures within the sample to minimize `t1` noise in multidimensional NMR experiments. This experiment measures the effect under typical conditions.

^{2}H Pulse and Decoupling Tests

The standard AutoTest sample has 99% D_2O, which can be directly observed using the lock circuitry in the probe. Setting tn=`lk` causes channel 1 to produce ^{2}H rf, which is then routed into the ^{2}H Diplexer box on the side of the magnet leg and then on to the lock BNC of the probe. This setup is used for ^{2}H gradient shimming. Since ^{2}H is observed, the pw90 can be determined at any power level. The normal channel 1 hardware setup is used, and because ^{2}H is a low-frequency nucleus, the rf is automatically routed through the first broadband amplifier before going to the rf relay at the magnet leg. If tn is not `lk`, the rf is routed to the broadband preamp for normal observe. With suitable recabling of rf from a decoupling channel, the ^{2}H pulses can be delivered from a decoupling channel, and thus calibrated. Only one cable change is necessary and this is described in the macros and pulse sequences used. The pw90's determined can be used to calculate dmf/dpwr values for heteronuclear decoupling experiments on the channel used. Results are stored in appropriate History files. Usually, only one particular channel is used for ^{2}H decoupling, so only one of the tests below is used for a particular spectrometer.

^{2}H pw90 using Channel 1

Experiment: The power level of channel one's attenuator is varied for a fixed pw of 150 µs. At the power giving a maximum signal, the pulse width is then varied to obtain a calibrated pw90. The power level and pw90 values are stored in the history file `H2PW90`.

^{2}H pw90 using Channel 3

Experiment: The power level of channel three's attenuator is varied for a fixed pw of 150 µs. At the power giving a maximum signal, the pulse width is varied to obtain a calibrated pw90. The power level and `pw90` values are stored in the history file `H2PW90_ch3`.

^{2}H *pw90 using Channel 4*

Experiment: The power level of channel four's attenuator is varied for a fixed pw of 150 µs. At the power giving a maximum signal, the pulse width is varied to obtain a calibrated pw90. The power level and `pw90` values are stored in the history file `H2PW90_ch43`.

^{2}H pw90 using Lock/Decoupler

Experiment: The power level of the Lock/Decoupler's attenuator is varied for a fixed pw of 150 µs. At the power giving a maximum signal, the pulse width is varied to obtain a calibrated pw90. The power level and `pw90` values are stored in the history file `H2PW90_lkdec`. The Lock/Decoupler is normally the last channel.

^{2}H 90° Pulse Amplitude Stability and Sensitivity using Channel 1.

Experiment: The ^{2}H D_2O signal is recorded following 90° pulses on channel 1. Average amplitude and standard deviation are recorded and stored in the `H2STAB90` history file. Sensitivity and linewidth are measured and stored in the `H2SENSITIVITY` history file.

Purpose: ^2H heteronuclear decoupling involves 90° pulses and constant amplitude, phase-modulated pulse trains. A stable amplitude is essential for optimum results. High sensitivity gives a more stable lock. This test permits tracking of lock sensitivity over time.

^2H 90° Pulse Amplitude Stability using Channel 3.

Experiment: The ^2H D_2O signal is recorded following 90° pulses on channel 3. Average amplitude and standard deviation are recorded and stored in the `H2STAB90ch3` history file.

Purpose: ^2H heteronuclear decoupling involves 90° pulses and constant amplitude, phase-modulated pulse trains. Stable amplitude is essential for optimum results.

^2H 90° Pulse Amplitude Stability using Channel 4.

Experiment: The ^2H D_2O signal is recorded following 90° pulses on channel 4. Average amplitude and standard deviation are recorded and stored in the `H2STAB90ch4` history file.

Purpose: ^2H heteronuclear decoupling involves 90° pulses and constant amplitude, phase-modulated pulse trains. Stable amplitude is essential for optimum results.

^2H 90° Pulse Amplitude Stability using Lock/Decoupler.

Experiment: The ^2H D_2O signal is recorded following 90° pulses using a Lock/Decoupler. Average amplitude and standard deviation are recorded and stored in the `H2STAB90lkdec` history file.

Purpose: ^2H heteronuclear decoupling involves 90° pulses and constant amplitude, phase-modulated pulse trains. Stable amplitude is essential for optimum results.

^2H Spinlock Test using Channel 1.

Experiment: The ^2H D_2O signal is recorded following a spinlock pulse on channel 1. Average amplitude and standard deviation are recorded and stored in the `H2SPNLKSTAB` history file.

Purpose: ^2H heteronuclear decoupling involves 90° pulses and constant amplitude, phase-modulated pulse trains. The lock circuitry should be able to provide a high-power spinlock.

Installation Tests for Cryogenic Probes

After a cryogenic probe is installed a series of experiments are performed to both condition the probe and to verify its operation. These experiments run a ^{13}C HSQC pulse sequence with combinations of ^{13}C and ^{15}N decoupling during acquisition. The experiments run for 15, 30, 60, or 90 minutes followed by a test in which the power of ^{13}C and ^{15}N decoupling is varied while noise in the FID is measured (no observe pulse is used). The results are plotted to determine the extent of any excess noise from decoupling. The test is usually repeated automatically until stopped by the user.

This test conditions the probe by heating the coil enough to drive off (mainly) absorbed water. This water vapor (in the vacuum chamber of the probe) can give plasma noise under an rf field which adds to the noise of the FID and degrades s/n ratios. As the probe conditions, the residual water is pumped away by the vacuum pump.

15-Minute Conditioning (ATcryo15)

Experiment: 15 minutes of conditioning is done followed by the noise test.

30-Minute Conditioning (ATcryo30)

Experiment: 30 minutes of conditioning is done followed by the noise test.

60-Minute Conditioning (ATcryo60)

Experiment: 60 minutes of conditioning is done followed by the noise test.

90-Minute Conditioning (ATcryo90)

Experiment: 90 minutes of conditioning is done followed by the noise test.

Other Test Descriptions

Heating During Spin Lock Test

Experiment: The same test as in the variable temperature test is performed using a 70-ms ^{1}H pulse at an rf field strength of 10 kHz with a total recycle time of 1.5 s, including acquisition. The data are plotted in the same way as in the VT test.

Purpose – Many modern experiments use spin locks or decoupling within their pulse sequences. Often, rf fields can cause significant sample heating. Depending on the nature of the probe, this heating can be a problem causing baseline artifacts and t_1 *noise*. It is important to quantify the amount of sample heating, the speed in attaining a new equilibrium temperature, the homogeneity of temperature throughout the sample during the heating period, and the final change in temperature.

This test imposes a rather strong (10 kHz) rf field for a period of time often used in TOCSY experiments, using a recycle time of 1.5 s (including acquisition). Single transients are acquired, at a rate of one per 1.5 s. The data show any temperature change at the expected 0.01 ppm/degree. The intensity and/or linewidth can be used to measure temperature homogeneity. The number of transients needed to attain a new equilibrium temperature measures the ability of the probe to stabilize the effects of internal sample heating. The final shift value indicates the total temperature change. This can be used to reduce the requested temperature value so as to obtain the desired equilibrium. Of course, the amount of heating is a function of the sample itself, primarily its salt content.

The same approach may be used to follow the actual temperature in the sample under the influence of X-nucleus decoupling.

Lock Tests

Experiment – Lock power is varied over a 30 dB range and the lock level recorded. The experiment is repeated for the lock gain. A log regression analysis is performed to confirm the relationship between the lock signal and power/gain.

Purpose of Lock Gain Test: Lock gain is selectable in a logarithmic manner (in dB). In an ideal case, variation of receiver gain should produce a logarithmic dependence of signal strength.

Purpose of Lock Power Control Using an RF Attenuator: Overall lock power control is accomplished using computer-controlled rf attenuators. The amplitude response should also be logarithmic. A log regression analysis should show the extent of fit to the ideal.

Spectral Purity Test

Experiment: Four single-scan 100 kHz spectral width spectra are acquired with no pulse. Each spectrum is plotted with a few millimeters of noise.

Purpose: RF purity of the transmitter and receiver can be tested by recording data without any excitation pulse. The spectrum reveals any artifactual signals.

Variable Temperature Test

Experiment: Single-scan spectra are acquired during an increase of 5°C in sample temperature. Spectra are recorded sequentially. Spectra are taken every 2 s until the sample reaches equilibrium, as reflected in a stable chemical shift of a methyl proton. Spectra are plotted in a stacked manner to permit examination of the rate of change of temperature, the homogeneity of temperature, and the length of time necessary to reach equilibrium.

Purpose: The sample temperature is increased by 5^{o}C while recording spectra every 2 s. Most methyl resonances show a chemical shift of `(sfrq/100)` Hz/^{o}C and this shift, therefore, indicates the actual temperature distribution within the sample. The methyl resonance should move quickly and homogeneously to its new equilibrium position. The rate of change and homogeneity of change demonstrate the VT performance of the probe and regulation hardware.

Small-Angle Phase Shift Test

Experiment: Single-scan spectra are acquired in which the phase of the pulse is incremented by 10° in each spectrum through a full 360° at constant receiver phase. Spectra are plotted in a horizontal stack to show a smooth phase rotation of the spectrum. The test is repeated using pulses generated from channel 2.

Purpose: Small-angle phase adjustment is used in multiple-quantum selection (`q>2`), phase-modulated pulses, and a variety of complex pulse sequences. This test exercises the phase-shift hardware by varying the pulse phase in small increments over 360°.

Tests Using Salty Sample

The following tests are designed for a standard AutoTest sample with added salt.

^{1}H RF Homogeneity Experiment: One hundred experiments are run in which the pulse width is incremented from 1 to 100 s. The spectra are plotted in a horizontal stack in phased mode and sufficiently expanded so that the base of the water can be examined using the same phase settings for each spectrum (using channel 1).

Pulse Stability

Experiment: A single-scan pulse experiment is repeated 20 times and the spectra plotted in a horizontal stack. The average peak amplitude, rms deviation and sensitivity are measured and reported. This test is run for the following:

- 90° flip pulses
- 30° flip pulses
- 10° flip pulses
- 1 µs pulses

Purpose: The same tests are performed as for the standard AutoTest sample, but for a sample with added salt. Any difference in performance is highly relevant for user samples containing salts or buffers, as these constituents detune the probe and result in lower sensitivity and longer pw90’s.

Heating During Spin Lock Test

Experiment: This test is performed using a 70-ms ^{1}H pulse at an rf field strength of 10 kHz with a total recycle time of 1.5 s, including acquisition. The presence of salt provides opportunity for additional sample heating and this test measures the extent of heating under the same conditions as used for the non-salty sample.

Decoupler Heating

The same test as in the variable temperature test is performed but this time using a 75-ms ^{13}C decoupling period prior to acquisition within a total recycle time of 1.5 s, including acquisition. One-hundred, single-scan spectra are collected with ^{13}C decoupling followed by 100 identical spectra with no decoupling. The spectra are plotted in a stacked manner to permit examination of the rate of change of temperature, the homogeneity of temperature, and the length of time necessary to reach equilibrium. The rf field strength is sufficient to decouple over a 160 ppm range using garp-1. The presence of salt provides opportunity for additional sample heating and this test measures the extent of heating under the same conditions as used for the non-salty sample.

Chapter 22. Magnet and Spectrometer Maintenance

Sections in this chapter:

- 22.1 "Preventative Maintenance," this page
- 22.2 "Handling Liquid Helium," page 318
- 22.3 "Continuing Dewar Service," page 323
- 22.4 "Troubleshooting," page 327

22.1 Preventative Maintenance

Performing preventative maintenance on a schedule can go a long way toward trouble-free operation of the spectrometer system. If problems do occur, we suggest you review the troubleshooting section in this chapter before calling your Varian service person. The following scheduled preventative maintenance and documentation is suggested.

Scheduled Maintenance

A maintenance schedule such as the following is recommended.

Weekly

1. Check air line traps for dirt or condensed water.
2. Record the following readings (note that the flow rates of nitrogen and helium depends on a variety of factors such as atmospheric pressure and a high reading is not necessarily an indication of a problem, but it is worth investigating):
   - Pressure at the air valve for the magnet.
   - Liquid nitrogen level.
   - Readings on the nitrogen and helium flow meters.

Twice Each Month

- Check liquid helium level.

Monthly or Bimonthly

- Check signal-to-noise and lineshape using the standard proton sample, the ^{13}C 90° pulse width, and the decoupler field strength. keep the resulting spectra and parameters in a secure place for future reference.

Periodically

Back up data to tape using programs such as `dump` and `tar`.

Maintenance Documentation

It is a good idea to maintain three notebooks to document your system:

- *System Log* – For entering cryogen check and fills, service calls, problems, etc.
- *Operations Log* – For recording what you do every day—what samples were run, what new procedures were tried, what macros were written, what problems were encountered, etc.
- *Procedures Log* –For documenting new procedures, for example, a successful DEPT run (listing all parameters), a standard parameter set for ^{11}B, the procedure for determining 90° pulse widths, etc.

Each of these notebooks should set aside space for a table of contents on the first page and contain as much detail as possible.

22.2 Handling Liquid Helium

Safe and economical use of liquid helium (LHe) requires close attention to details. This section examines the physical properties of helium to show the relationship of these properties to problems in transferring helium.

WARNING: **To prevent possible personal injury, observe all warnings posted on equipment and stated in this manual. Before operating or servicing any part of the system, read the “Safety Precautions” section in the front of this manual.**

Physical Properties of Helium

Table 18 compares the physical properties of helium to nitrogen. When establishing general procedures for handling LHe, consider the special properties of helium described in the following sections.

Table 18. Physical Properties of Helium and Nitrogen

Property	*Helium*	*Nitrogen*
Molecular weight	4.0026	28.01
Boiling point (1 atm)	4.2 K	77.4 K
Melting point	1.1 K (25 atm)	63.2 K (1 atm)
Critical temperature	5.2 K	126.0 K
Critical pressure	2.26 atm	33.5 atm
Density of liquid at boiling point	0.125 gm/cm^3	0.807 gm/cm^3
Density of gas at boiling point	0.0176 gm/cm^3	0.00462 gm/cm^3
Density of gas at 27°C	0.163 kg/m^3	1.138 kg/m^3
Heat of vaporization	2.58 J/cm^3	161.0 J/cm^3
Δ H=H300-H boiling point	192.0 J/cm^3 liq.	186.0 J/cm^3 liq.

Sources: R. B. Scott, 1959, *Cryogen Engineering*, D. Van Nostraud Co., Inc.
R. Barron, 1966, *Cryogenic Systems*, McGraw Hill Book Co.
A Wexler, 1951, *Journal of Applied Physics*, 22:1463.

Low Boiling Point

Because LHe at atmospheric pressure boils at 4.2 K (kelvin), efficient insulation is required to minimize heat input to the cryogenic fluid (4.2 K is equivalent to –269.0°C). Care must be taken to prevent frozen air plugs from blocking pressure-relief valves and to keep other gases from entering the LHe container. Storage at a pressure slightly above atmospheric is essential to exclude air from storage containers and magnet dewars containing LHe.

Low Latent Heat of Vaporization

Of all cryogens, helium has the lowest heat of vaporization, on either a weight or volume basis. On a volume basis, the latent heat of helium is about one-sixtieth the latent heat of nitrogen and one-half the latent heat of liquid hydrogen. Therefore, the slightest heat input from any source (pressurizing gas or poor insulation) causes loss of LHe.

High Specific Heat of Vapor

To vaporize one liter of LHe requires 2580 joules. The same heat input will raise the temperature of the resultant vapor less than 4 K. To heat the vapor to room temperature requires 1.9×10^5 joules. Liquid losses can be substantially reduced, including loses from initial system cooldown, by using this vaporization property. Disregard of the property can lead to high losses during operations, such as topping-off a partially filled container.

Low Liquid Density

The density of LHe, 125 grams per liter, is about one-sixth the density of liquid nitrogen (LN). Because of the low density, the LHe content of a nitrogen-shielded container is difficult to gauge by weight.

High Vapor Density

Although helium is a very light gas at normal pressure and temperature, its vapor density at the normal boiling point is higher than that of any of the atmospheric gases. Helium vapor expands greatly upon warming to ambient temperatures, and if confined to a fixed volume, the pressure increase is correspondingly large. As a result, “empty” containers, filled only with very cold vapor, must be treated the same as containers containing liquid. Adequate venting must be maintained, and precautions must be taken to prevent formation of frozen air plugs in the container neck tubes.

Small Difference Between Vapor and Liquid Densities

The small difference between the densities of LHe and cold helium vapor may lead to high entrainment losses under flow or boiling conditions.

Handling Liquid Helium Safely

WARNING: **The extremely low temperature of liquefied helium and nitrogen can cause skin damage similar to high-temperature burns. Contact with the cold gas evolving from the liquid may produce the same effect. Delicate body tissues, such as the eyes, are easily damaged by exposure to cold gas or liquid. Skin can stick to metal that is refrigerated by liquid helium and can tear when pulled away. Immediately flood with large quantities of unheated water any area of the body that is “burned” by liquid or cold gas, and then apply cold**

compresses. If the skin is blistered or there is any chance the eyes are affected, immediately seek medical treatment.

Before any transfer of LHe is attempted, know the safety precautions and operating instructions for the hardware used, especially the following:

- Avoid contact of gaseous helium with any part of the body. Wear safety coverings, including a complete face mask and thermo-insulated gloves. The gloves should be clean and dry, and should be loose-fitting so that you can throw them off quickly if frozen by contact with the gas.
- Handle and store helium containers in adequately ventilated areas. Helium and nitrogen gases are not toxic or flammable. However, gas evolving from the liquid in an enclosed space can reduce the oxygen content of the surrounding air and cause a potential asphyxiation hazard. Because nitrogen and helium gases are odorless, colorless, and tasteless, their presence is undetectable by the human senses.
- Be sure pressure-relief valves are adequately sized. LHe vaporizes rapidly when heat is introduced. Therefore, the pressure-relief valve for LHe containers and equipment must have sufficient capacity to safely release the rapidly expanding helium gas.
- Prevent pressure-relief valves from freezing open. An open valve allows the container to blow down, and eventually air backflows into the container and freezes.
- Neck tubes may still be unobstructed even if pressure inside the helium container is above atmospheric. Check containers when received and periodically recheck each container to be sure the vents are unobstructed.
- Very cold helium gas vents upon removal of the top fitting of a container. Cap or seal the container rapidly because the top fillings and valves will become cold and frosted, making sealing more difficult.

Most LHe storage containers are designed with an inner assembly within the vacuum jacket. Because the innermost neck is made of a thin material, the container is vulnerable to rough handling, and especially to a sudden shock. Take the following precautions to prevent damaging containers:

- Never try to pour liquid out of the container; use a vacuum-insulated transfer tube.
- Keep the container vertical at all times.
- Never roll, tip or slide the container. Use a dolly when moving it.
- Do not drop the container.
- When transporting the container, fasten it securely to prevent it from moving or falling over.

Measuring Liquid Helium

The thermal acoustical liquid level indicator, or flutter tube, is used to locate the surface of LHe inside a container. The device depends upon the thermal-acoustic oscillations generated in tubes that have one end at room temperature and the other at LHe temperature.

The flutter tube consists of a length of 1/8 in. thin-wall stainless steel tubing with a small cup, or funnel, shape at the warm end, flared to a diameter of about 1/2 in. When the tubing end is inserted into a dewar of LHe, acoustical oscillations occur that abruptly change in intensity and frequency as the end of the tube passes through the liquid zone into the dense gas zone.

Using the Flutter Tube

Take the following steps to measure LHe with a flutter tube:

1. Slowly insert the tube into the LHe container until the tube touches bottom, then place an alligator clamp on the tube, level with the top of the container.
2. While slowly raising the tube, observe vibrations by closing off the top end of the tube with your thumb. When the end of the probe passes from the liquid zone to the dense gas zone, the vibrations intensify and the frequency changes. Place another clamp on the tube, level with the top of the container.
3. Measure the distance between centers of the two clamps; this is the LHe depth.

Air and Moisture Ice Hazard

A hazard is created when air and moisture are pulled into the nitrogen shield and the inner chamber of LHe containers. Sudden barometric pressure changes can cause the air diffusion to increase. Once into the LHe neck tube, the heavier gases in the air diffuse to the bottom and condense, and then solidify near the LHe end of the tubes. The moisture in the air also forms ice in the LN tubes. The plugs formed by the ice can cause damage to equipment.

CAUTION: **Air and moisture condensation can result in a solid ice plug in the LHe chamber pressure vents. This obstruction can result in damaged equipment. Keep all vents capped or sealed when possible.**

Transferring Liquid Helium

The following sections summarize transferring LHe from a storage container to the LHe chamber in a superconducting magnet. The actual procedure used to transfer LHe when installing and maintaining Varian NMR superconducting magnets is given later in this chapter.

Helium Movement through the Transfer Line

When a cryogenic liquid is started through a typical transfer line that is initially at room temperature, the liquid at first is quickly evaporated and nearly the entire line contains only gas. As more liquid enters, part of the cryogenic source end of the line becomes cooled below the saturation temperature, and this part contains a pure liquid phase. Toward the destination end of the transfer line is a region in which both liquid and vapor are present. In the remainder of the transfer line, only gas is flowing. A very light gas plume appears at the exit end of the tube.

As the line is further cooled by the evaporating liquid and by the resulting cold vapor, the liquid phase travels farther along the line until finally liquid approaches the exit end. The vaporized liquid appears as a very dense white fog blown out from the end of the tube.

If the warm vapor that precedes the liquid is discharged into the LHe chamber of the magnet, it evaporates some of the liquid already present. This evaporation can be largely avoided by making sure the transfer tube is precooled before inserting it into the equipment, which already contains LHe.

If the warm line is permitted to deliver excessive warm gaseous helium to the LHe chamber of the magnet, a serious evaporation of liquid can result. The turbulence caused by the discharge of warm (nonliquid) helium can result in a quench of the magnet field present.

Preparations for Transfer

The transfer tube must have a good vacuum (50 microns warm, maximum pressure). The appearance of condensed moisture or frost on the tube during transfer shows that the vacuum is poor, reducing the insulation properties created by the vacuum and possibly preventing the transfer. A small spot of frost indicates a thermal “short” caused by contact of metal surfaces. Stop the transfer immediately because LHe is vaporizing in the tube, due to the heat conducted through this thermal short.

Make sure that the vacuum-jacketed portion of the transfer tube extends below the bottom of the neck tubes on the supply and receiving containers.

Typical Transfer Procedure

1. Verify the amount of LHe needed and make sure that amount is available for transfer.
2. Set or verify that helium gas pressure is 4 to 5 psig.
3. Precool the transfer tube, as follows:
    a. Vent the storage dewar.
    b. Insert the transfer tube in the storage dewar only.
    c. Reduce the boiloff pressure by venting.
    d. Close the low-pressure safety vent valve on the storage dewar.
    e. Pressurize the dewar. Set flow to 4 on the flowmeter (pressure will slowly rise). Wait for the dense, white, “flame-like” exit gas from the deflector nozzle (while waiting, release the internal pressure from the magnet slowly). Vent the storage dewar pressure (leave dewar vent open to prevent pressure buildup).
4. Insert the transfer tube and repressurize the storage dewar, as follows:
    a. Wipe to clear any ice build up at the deflector nozzle.
    b. Open the magnet entry port.
    c. Lift up and insert the transfer tube into both dewar and magnet (lower to the insertion stop located on the transfer tube magnet side).
    d. Close the dewar vent (the flowmeter should still be set to 5). Pressure should slowly rise.
5. Start the fill. While filling, pressure in the dewar should slowly rise.
6. When the exit cloud is accelerated and shows increased density, filling is complete. Complete the transfer as follows:
    a. Vent the storage dewar pressure.
    b. Remove the transfer tube from both the dewar and the magnet.
    c. Close and cap the storage dewar and the magnet.
    d. Open the low-pressure safety valve on the storage dewar.
    e. Close off the helium gas source.

22.3 Continuing Dewar Service

This section describes servicing a dewar with liquid helium after the liquids have dropped to the point where refilling (topping off) is necessary. Because of its very low latent heat of vaporization, LHe must be transferred through a well-insulated vacuum jacketed tube.

WARNING: **Before attempting any transfer of liquid helium or liquid nitrogen, know the safety precautions and operating instructions for the hardware in use. Serious injury can occur in the handling of very cold gases and liquids.**

Liquid Helium Service

Before the transfer occurs, *the transfer tube must be precooled*. Precooling takes from 10 to 40 seconds. Immediately after precooling, the transfer tube is inserted into the magnet dewar and the transfer of LHe initiated. Difficulties in filling occur if unnecessary delays occur between precooling and insertion of the transfer tube.

CAUTION: **When servicing with helium, do not bend or twist the transfer tube when inside the magnet neck tube because the tube is thin-wall stainless steel and is easily punctured with sharp objects. When clearing ice plugs, use the supplied clean-out tube.**

1. Position the LHe storage container with the discharge port about 26.5 in. (68 cm) horizontally from the HELIUM SERVICE port, so that the transfer tube can be inserted into the storage container and the magnet HELIUM SERVICE port simultaneously. (This distance applies only to the LHe transfer tube available from Varian.)
2. Make sure that the helium gas storage cylinder is equipped with a pressure regulator having a range of 2 to 10 psig.
3. *Measure the amount of LHe in the storage container and in the magnet dewar* to be sure of a sufficient supply to top off the magnet. Attempting to top off the dewar with a minimal supply could result in a quench. The available liquid helium in the storage container is not the capacity of the storage dewar or the amount of liquid helium in the dewar upon its delivery. *The available liquid helium is the amount of liquid helium accessible to the transfer tube with a stinger.*
4. Make sure, by measurement, that the tip of the transfer tube (that is, the end of the stinger if used) on the storage container side reaches to within 2 cm of the bottom of the dewar when the stop tab on the magnet side of the transfer tube is resting on the lip of the magnet HELIUM SERVICE port. Check that the deflector nozzle is attached to the magnet end of the transfer tube.
5. Depressurize the storage container. During shipment and non-use time, the storage dewar may build up internal pressure.
6. Place the pressurizing collar on the storage container side of the transfer tube.
7. *Precool the helium transfer tube* by slowly inserting one end into the storage dewar with the other end open to the room (see Figure 35). Do not close the storage dewar exhaust port during insertion. The heat of the transfer tube (room temperature) will cause increased liquid helium boiloff while being inserted. This helium boiloff should *not* be allowed to build up inside the storage dewar causing pressure.

   Once the transfer tube is fully inserted, pressurize the storage dewar by closing the exhaust port. Adjust the flowmeter valve so that the flow rate is about 5 on the scale.

The pressure will rise slowly. After 10 to 40 seconds, a plume of LHe (in the form of a very dense, milk-like white, flamed-shaped fog) emerges from the deflector nozzle, indicating that the precooling has been accomplished.

8. After observing the exhaust for 5 to 15 seconds, depressurize the storage dewar by opening the exhaust port.

   If the transfer tube frosts over or condenses moisture from the room air (in areas not directly exposed to cold exhaust), it is likely that there is a poor or faulty vacuum within the transfer tube jacket. The transfer tube should be removed from use, warmed, and pumped down to 5 microns (5 millitorr) or better.

9. Pull the transfer tube up enough to clear the magnet dewar. Remove the HELIUM SERVICE filling port cap. Swing the transfer tube over the magnet and immediately insert it slowly into the magnet LHe access port and the storage container simultaneously (see Figure 36). As a guide, use 10 to 15 seconds for insertion up to the stop tab.

   If you observe excessive turbulence or a sudden increase in the exhaust rate, the transfer tube insertion is too rapid and hot gas is reaching the LHe in the magnet dewar. This can cause a quench when the LHe level is below the top of the solenoid. The insertion rate should be slowed down.

10. Once the transfer tube is fully inserted so that the stop disk rests on the lip of the magnet access port, start to pressurize the storage dewar by closing the storage container exhaust port. The flow rate of approximately 5 on the scale is still present and will now pressurize the dewar.

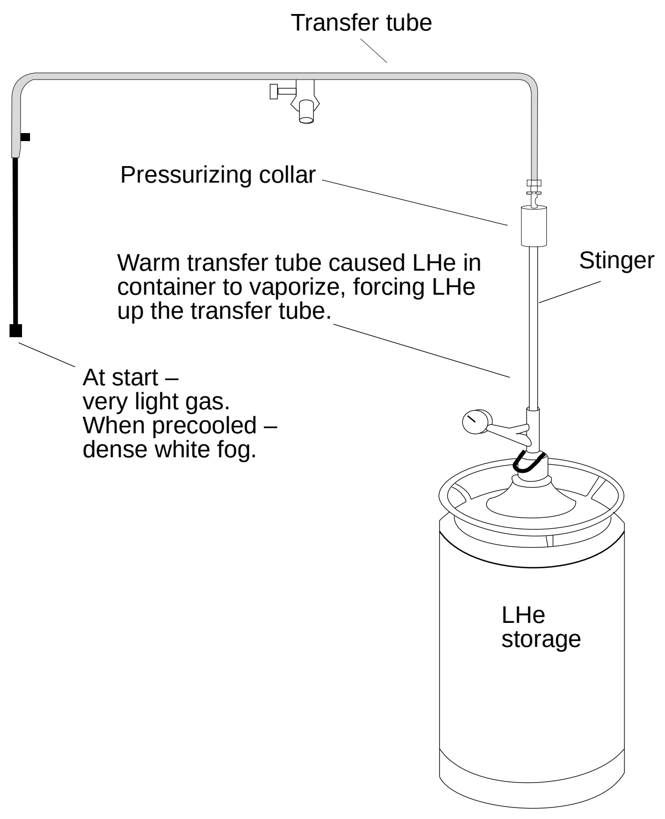

Transfer tube
Pressurizing collar
Warm transfer tube caused LHe in container to vaporize, forcing LHe up the transfer tube.
Stinger
At start – very light gas. When precooled – dense white fog.
LHe storage

Figure 35. Precooling the Liquid Helium Transfer Tube

CAUTION: **Watch the pressure gauge on the gauge head assembly. Any rapid pressure buildup may indicate a frozen air or water blockage in the transfer tube or the fill tube. If ice blockage occurs, halt the LHe transfer, remove the LHe transfer line, and again purge the line with gaseous helium. After the line is cleared, restart the transfer from the beginning. Be sure to plug the storage container and the HELIUM SERVICE port as soon as the LHe transfer tube is removed from them in order to prevent air and moisture from entering the dewars.**

Open the vent port of the magnet on the filling side only.

The pressure will rise slowly toward 1 psig. During the 15 seconds to 1 minute, a collapsing exhaust plume is usually observed. The collapsing exhaust indicates that the liquid helium is reaching the magnet end of the transfer tube and is filling the magnet.

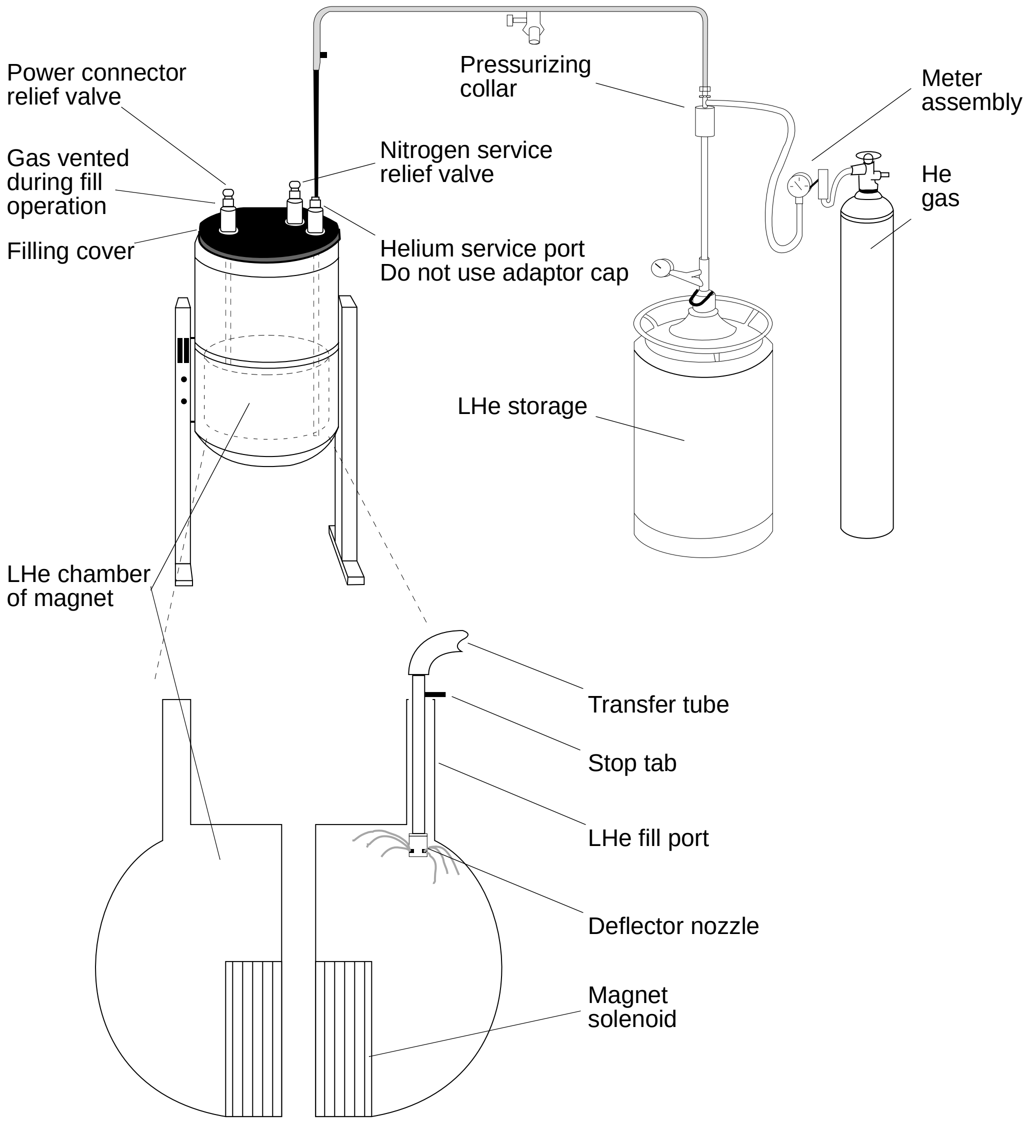


Figure 36. Routine Filling of the Magnet Dewar

If the pressure is not increasing toward 1 psig when the flowrate is set to 5 on the scale, there may be a leak of the pressurizing gas that is preventing the storage dewar from being pressurized.

If the pressure exceeds 2 psig when the flowrate is set to 5, it is likely a blockage has occurred in the transfer tube. It will be necessary to remove the transfer tube, cap the magnet, warm the transfer tube, and start over again at the precooling step.

If a frost spot occurs in any section of the transfer tube that is not directly exposed to cold exhaust, there is probably a metal to metal touch across the vacuum jacket. The transfer tube should be replaced. A defective transfer tube can cause a magnet quench. Varian recommends that you observe the transfer tube for normal operation at each use. The transfer tube can be routinely pumped down once or twice a year as preventive maintenance.

11. The filling continues at this pressure and rate until the magnet dewar is full, which is indicated by a marked increase in the rate of exhaust from the magnet access port. This increased exhaust is accompanied by an increase in the density of the exhaust cloud. If there is any doubt about the dewar contents level, use a flutter tube measurement to verify.
12. Depressurize the storage dewar. Remove the transfer tube from both containers. (Removal is much less risky than insertion.) Cap the magnet access port. Close the appropriate storage dewar valves.

Helium Refilling on Oxford 200- and 300-MHz Magnets

To maintain optimum refill safety and minimize the risk of accidental quenches, follow these special recommendations when refilling Oxford 200- and 300-MHz magnets:

- When you start filling the magnet (particularly if the helium level is low), open *only* the vent port on the side on which the transfer tube is inserted. Although this procedure should fill most magnets completely, some magnets are only partially filled.
- If a magnet is only partially filled with the vent port open, wait several minutes until the transfer has stabilized (indicated by the collapsing of the initial plume), and then open the port on the demountable lead side. If the second port is opened too soon after filling with the vent port open, warm gases could be sent across the solenoid.
- Under no circumstances should LHe be transferred with the port on the transfer tube side closed and the port on the demountable lead side open.

Determining Stinger Lengths for Storage Containers

A stinger may be necessary to extend the transfer tube into the LHe storage container. Refer to Figure 37 for the measurements needed to determine the stinger length for a storage container. Take the following steps for each container:

1. Measure distance A, the top of the storage container storage port to the bottom of the liquid helium chamber (a flutter tube can be used for this purpose).
2. Measure distance B, the top of the storage container service port to the top of the liquid helium chamber (a stiff wire with a crimp on the bottom to catch at the top of the chamber works well).
3. Subtract B from A to obtain the depth of the liquid helium chamber. This is measurement C, *the maximum length of a stinger for this container*.

CAUTION: **The stinger on the storage container side of the transfer tube must not be longer than the height of the LHe chamber (distance C). Exceeding this maximum will cause filling difficulties and can cause a quench.**

The following steps identify whether an elevating platform is needed below the storage dewar. Because the stinger has a limit to its maximum length, and the transfer tube cannot be lowered further than the stop disk allows (on the magnet side), an elevating platform for the storage dewar may be required to enable the stinger entry tip to reach the LHe to be transferred.

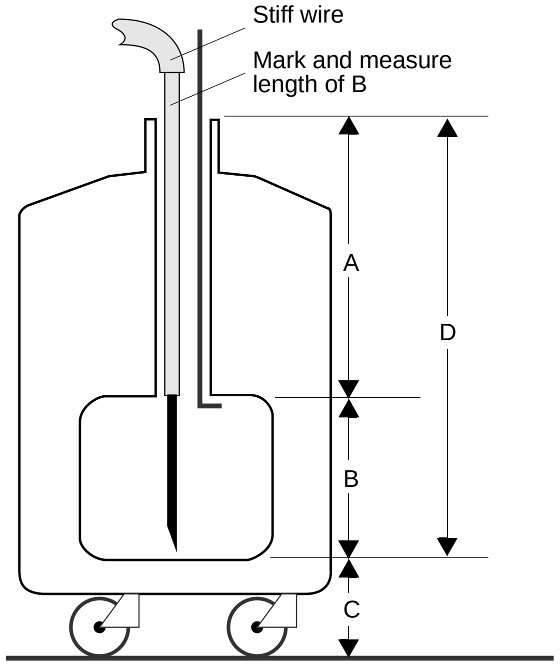


Figure 37. Stinger Configuration Measurements on the Storage Container

1. Measure distance D, the bottom of the liquid helium chamber to the floor.
2. Add C to D to give the distance from the top of the LHe chamber to the floor.

- If this distance is 26 inches or greater, then you will be able to fill from this container without an additional platform to elevate the storage dewar (this distance applies only to the LHe transfer tube available from Varian.).
- If this distance is less than 26 inches, constructing a platform under the storage container is necessary before proceeding. The platform must be designed to elevate the container so that the storage chamber top (C + D in Figure 37) is level with or higher than 26 inches above floor level. The design of the platform must also allow the storage container to be positioned with its service port 68 cm (26 3/4 in.) from the HELIUM SERVICE port on the magnet, as measured on a horizontal line that simulates the transfer tube.

22.4 Troubleshooting

Sample Scoring Problems

Sample scoring and spinning instabilities are often caused by improper mating between the upper barrel and the probe. The following spin test checks for this problem.

1. Cover the sample tube using a felt-tipped ink marker pen (blue or black ink is best).
2. Let the sample spin for 1 to 3 hours in the magnet.

Any scoring should show up clearly in the inked sample tube. If scoring is evident, perform the following:

1. Loosen the probe flange thumbscrews one full turn and retighten them without twisting the probe body.
2. Spin test the sample again.

If scoring is still present, follow this procedure:

1. Loosen the upper barrel thumbscrews one full turn.
2. Loosen the probe flange thumbscrews one full turn.
3. Semi-loosen the probe flange set screws.
4. Push up on the probe body until the upper barrel moves approximately 1 inch.
5. Push down on the upper barrel until it hits the stops.
6. Tighten the probe flange set screws and thumbscrews.
7. Tighten the upper barrel thumbscrews.
8. Insert the sample and spin test for scoring.

If scoring is still evident, perform steps 1 to 8 again. If the scoring continues, contact you field service engineer.

Sample Changer Troubleshooting

Some problems that can be dealt with by the user are the following:

Sample Changer Not Responding

On the SMS sample changer, first check that the emergency stop button is off, then check the power switches on the back of the System V control and the robot. Also check the cable from System V control to the NMR console.

On the ASM-100 sample changer, check the power switch on lower rear of sample changer, check that RS-232 cable to sample changer is connected, and check whether any fuses in the sample changer are blown.

Gripper Drops Sample Back into Magnet when Sample is Ejected

On the SMS sample changer, check for bent fingers.

On the ASM-100 sample changer, gripper is not grasping enough of spinner— raise air pressure to sample changer by 1 to 2 pounds, or adjust the height of the gripper retrieval (bottom) position.

Other Troubleshooting Solutions

Spinner does not rotate.

Remove, disassemble, and clean upper barrel with ethanol.

Magnet has high boil off.

Check for ice in the power connector.

Annotated Bibliography

This bibliography is organized into the following sections:

- "Sun and UNIX Newcomers," next
- "UNIX in General," next
- "Specific UNIX Tools," page 330
- "Networking," page 331
- "Network and UNIX Security," page 332
- "Internet and WWW," page 332
- "C Programming," page 333
- "UNIX Programmers," page 333
- "X Window System," page 334

An asterisk (*) indicates a book title that is particularly recommended.

Sun and UNIX Newcomers

The documentation that comes with the Sun workstation gives a brief introduction into the basics of the Solaris user interface. For most sites, the full UNIX documentation (which has to be ordered separately) is not required. Before buying the full documentation, you should first consider loading the UNIX online manuals (when loading UNIX), which will answer most questions.

UNIX in General

*Becker, G., M.S.E. Morris, and K.Slattery, *Solaris Implementation: A Guide for System Administrators*, Prentice Hall, Englewood Cliffs, NJ, 1995. (300 pp.) ISBN 0-13-353350-6

> A practical guide to UNIX administrators that migrate their system from SunOS 4.x to Solaris 2.x (up to Solaris 2.4). Covers Solaris installation, boot and shutdown files, system configuration using the AdminTool *and* using command line operations, network setup and maintenance, security setup and maintenance, managing software packages and Solaris patches, disk utilities and archiving procedures, controlling run states with boot files, using unbundled software products such as WABI.

Bourne, S.R., *The UNIX System*, Addison-Wesley Publishing Company, Wokingham, 1983. (351 pp.) ISBN 0-201-13791-7

> A practical guide to UNIX that describes basic UNIX operation, the text editors, the Bourne shell, the C programming language, UNIX system programming, `nroff` and `troff`, and the most important data manipulation tools. It does not include some of the 4.2 BSD enhancements, like the C shell.

*Gilly, D., *UNIX in a Nutshell, A Desktop Quick Reference for System V & Solaris 2.0*, O'Reilly & Associates Inc., 1992. (444 pp.) ISBN 1-56592-001-5

> A nice, compact quick reference book on System V (SVR4) UNIX and Solaris 2.0. It covers all important user commands in an alphabetically sorted command reference

section (about one-third of the book). Other chapters deal with the different shells, text editing (`emacs`, `vi`, `ex`, `sed`, `awk`), text formatting (`nroff`, `troff`), software development (`sccs`, `rcs`, `make`, `debugging`). System administration is not covered, and `nroff` and `troff` are becoming obsolete, but several subjects (`emacs`, `sccs`, `rcs`, `make`) are included that are barely covered in many larger books.

Libes, D. and S. Ressler, *Life with UNIX*, Prentice-Hall, 1989. (346 pp.) ISBN 0-13-536657-7

Interesting and often amusing book of UNIX trivia, history, etc. Contains lots of information difficult to find elsewhere. Probably for UNIX fans only.

**Nemeth, E., G. Snyder, S. Seebass, and T. Hein, *UNIX System Administration Handbook, 2nd Edition*, Prentice-Hall, 1995. ISBN 0-13-151051-7

Excellent book for everybody involved with UNIX system administration, Contains all the essentials, very easy to read. Includes networking, mail, `uucp`, and many helpful sample scripts for system administration. Almost as good (it's not a life course!) and even broader than administration training at Sun (and much, much cheaper, too!).

*Pew, J.A., *Guide to Solaris,* Ziff-Davis Press, 1993. (625 pp.), ISBN 1-56276-087-4

This is the official, SunSoft-approved learning tool for users of Solaris 2.1, with extensive chapters on the OpenLook, DeskSet, and DeskSet tools (file manager, shell tools, mailtool, calendar manager, etc.), separate chapters on command line UNIX covering the fundamentals, `vi` editor, C and Korn shells, networking (limited), and an outline of UNIX system administration (based on DeskSet system administration tools). Very readable, but for a better coverage of system administration see J. Winsor's *Solaris System Administrator's Guide* below.

*Winsor, J. *Solaris System Administrator's Guide*, *Second Edition*, Sun Microsystems Press, 1997. (335 pp.) ISBN 1-57870-040-X

Very useful book for system administrators that use Solaris 2.x and especially those who plan switching from Solaris 1.x (SunOS 4.1.x) to Solaris 2.x. Covers all important aspects of system administration that have undergone changes with Solaris 2.x, such as file system administration, networking, printing, troubleshooting, and new utilities (the Korn shell, system administrator tools, etc.). The second edition includes coverage of Solaris 2.6.

*Winsor, J. *Solaris Advanced System Administrator's Guide*, Ziff-Davis Press, 1994. (477 pp.) ISBN 1-56276-131-5

Complements the *Solaris System Administrator's Guide* by the same author. Covers advanced system administration topics such as installing mail services, adding NIS+ clients, the automounter, the service access facility for setting up printers, modems and terminals, and an introduction to Bourne shell programming, including shell scripts for automating routing administration tasks.

Specific UNIX Tools

Aho, A., B. Kernighan, and P. Weinberger, *The awk Programming Language*, Addison-Wesley, 1988. (210 pp.) ISBN 0-201-07981-X

The definitive book on *awk,* by its designers.

Anderson, G., and P. Anderson, *The UNIX C Shell Field Guide,* Prentice-Hall, Englewood Cliffs (New Jersey), 1986. (374 pp.) ISBN 0-13-937468-X

Very good, in-depth textbook on the C shell, one of the main features of 4.2 BSD UNIX not described in the book by S.R. Bourne. For people who want to go into extensive C shell programming.

Hansen, A., *vi—The UNIX Screen Editor, A User's Guide*, Prentice-Hall, New York, 1986. (230 pp.) ISBN 0-89303-928-4

Probably the most complete documentation on *vi*, only for people that intend to use *vi* almost full time.

Harrison, M., and M. McLennan., *Effective Tcl/Tk Programming, Writing Better Programs with Tcl and Tk*, Addison-Wesley, Reading (Mass.), 1998. (405 pp.) ISBN 0-201-63474-0

Nice introduction into Tcl/Tk programming; easy to read, well structured. Covers also newer features, such as grid, but not as detailed and exhaustive in general as the other two books mentioned below.

Ousterhout, J.K., *Tcl and the Tk Toolkit*, Addison-Wesley, Reading (Mass.), 1986. (458 pp.) ISBN 0-201-63337-X

A must for users who want to know more about Tcl/TK programming and who want to start writing their own visual tools. This is *the* reference book on the Tcl/Tk scripting language, written by the creator of the language. Complete, detailed, and exhaustive, although no longer quite up-to-date with the newest standards of the language.

Welch, B.B., *Practical Programming in Tcl and Tk, Second Edition*, Prentice-Hall PTR, Upper Saddle River (NJ.), 1997. (613 pp.) ISBN 0-13-616830-2

Very detailed and complete book, covers the newest level (8.0) of the Tcl/Tk language; definitely more up-to-date than the book by Ousterhout (see above), but not quite as well written. A complete language reference.

Networking

Costales, B., E. Allman, and N. Rickert, *sendmail*, O'Reilly & Associates Inc., 1993. (792 pp.) ISBN 1-56592-056-2

The `sendmail` "bible." One of the authors is the creator of `sendmail`. Contains all the information you need to fully understand and configure `sendmail` on your systems. Very detailed and complete. For network administrators only.

Hunt. C., *TCP/IP Network Administration*, O'Reilly & Associates Inc., 1994. (472 pp.) ISBN 0-937175-82-X

A detailed handbook about the internals of TCP/IP networking, covering the basics, configuration/setup, routing, DNS setup, sendmail, troubleshooting, security. For network administrators only.

Ravin, E., T. O'Reilly, D. Dougherty, and G. Todino, *Using & Managing UUCP*, O'Reilly & Associates Inc., 1996. (416 pp.) ISBN 0-56592-153-4

Excellent handbook on networking via dial-up connections. Covers most current versions of UUCP, and all the commands involved in communicating via UUCP.

Ramsey. R., *All About Administering NIS+*, Prentice Hall, 1994. (451 pp.) ISBN 0-13-309576-2

A detailed book that gives information on how NIS+ works, how to plan and set up the NIS+ service, transitioning from NIS to NIS+. For administrators of NIS+-based networks only.

Tanenbaum, A.S., *Computer Networks*, Prentice Hall, 1989. (658 pp.) ISBN 0-13-162959-X

The bible on the OSI networking model. Covers all aspects (including programming) of the OSI 7-layer networking model (one detailed chapter per layer), network topologies, and a complete annotated bibliography. For network administrators and designers and programmers of networking software.

Network and UNIX Security

Cheswick, W.R., and S.M. Bellovin, *Firewalls and Internet Security,* Addison-Wesley., 1994. (306 pp.) ISBN 0-201-63357-4

Well-written book. Gives an overview of common security holes with the various network protocols (including newer ones, like WWW) and describes the firewall functionality in detail. This book will enable gateway administrators to properly configure a firewall gateway, using a commercial firewall package or even by building a firewall. For specialists and gateway administrators only.

Garfinkel, S., and G. Spafford, *Practical UNIX & Internet Security*, O'Reilly & Associates Inc., 1996. (1004 pp.) ISBN 0-56592-148-8

A well-written, complete book on all aspects of UNIX security. Covers basic security features, communications (`uucp`, NFS, Kerberos and secure RPC), firewall machines, detecting and handling security incidents, data encryption, etc.

Internet and WWW

*Estrada, S., *Connecting to the Internet*, O'Reilly & Associates Inc., 1993. (170 pp.) ISBN 1-56592-061-9

A nice little guide into the basics of how to connect to the Internet. Discusses Network performance, Internet options, choosing a service provider (ISP), dialup and leased line connections. The book also contains a list of ISPs for most parts of the world. Quite useful if you want to set up a new Internet connection, at home or at work.

*Fox, D., and T. Downing, *HTML Web Publisher's Construction Kit*, Waite Group Press, 1995. (673 pp.) ISBN 1-57169-018-2

Nicely written and readable book. The first part discusses the most important Web browsers (Lynx, NCSA Mosaic, Netscape Navigator and others). The second part is a good guide about constructing a Web page, including an introduction to HTML (up to HTML 3), text formatting, graphics, forms, a brief introduction into CGI scripts, conversion of data into HTML format, and HTML editors. A third part deals with setting up a Web server. You obviously only need this book if you are into creating your own Web page.

*Krol, E., *The Whole Internet, User's Guide and Catalog*, O'Reilly & Associates Inc., 1994. (574 pp.) ISBN 1-56592-063-5

Instructive and complete book that explains all the services available on the Internet, telnet, FTP, News, Archie, Usenet, Gopher, WAIS, WWW, X software, etc. Includes a catalog of Internet resources and entry points for a wide range of topics.

C Programming

Brown, D.L., *From Pascal to C, An Introduction to the C Programming Language*, Wadsworth Publishing Company, Belmont (California), 1985. (151 pp.) ISBN 0-534-04602-9

More introductory than an in-depth textbook and not very complete, but still good for people who started programming in Pascal and now want to switch to C.

Harbison, S., and G. Steele, *C: A Reference Manual*, Prentice-Hall, 1987. (404 pp.) ISBN 0-13-109802-0

Excellent reference book for the C programming language, including the draft ANSI C. Much better than Kernighan and Ritchie as reference text.

Kernighan, B.W., and D.M. Ritchie, *The C Programming Language, Second Edition*, Prentice-Hall, Englewood Cliffs (New Jersey), 1988. (272 pp.) ISBN 0-13-110362-8

The "C bible"—the original textbook on C, written by the inventors of the language. Mostly still up to date and a must for most C programmers, although not particularly good for learning C. The second edition has all the code written in ANSI C. As a reference manual, the book by Harbison & Steele is much better.

Koenig, A., *C Traps and Pitfall*s, Addison-Wesley, 1989. (147 pp.) ISBN 0-201-17928-8

Very useful book. It discusses many of the subtle and not so subtle errors that C programmers make at some time.

van Wyk, Ch, *Data Structures and C Programs*, Addison-Wesley, 1988. (387 pp.) ISBN 0-201-16116-8

Typical computer science book of algorithms and data structures. The code examples are very clear and readable, generally rare for books on data structures written in C (most books on data structures are coded in Pascal, which is generally more readable, even for C programmers).

Wortman, L., and T. Sidebottom, *The C Programming Tutor*, Prentice-Hall, 1984. (274 pp.) ISBN 0-13-110024-6

Intensive introduction to C for competent programmers. Good introduction on pointers and some reasonably long (500 to 1000 lines) and useful sample programs.

UNIX Programmers

Bach, M., *The Design of the UNIX Operating System*, Prentice-Hall, 1986. (471 pp.) ISBN 0-13-201757-1

Excellent book about the internals of UNIX, mostly AT&T System V. Lots of pseudocode and C programs describing the internal data structures and algorithms. For UNIX maniacs only.

Kernighan, B., and R. Pike, *The UNIX Programming Environment*, Prentice-Hall, 1984. (357 pp.) ISBN 0-13-937681-X

A wide-ranging introduction to UNIX, intended more for programmers.

Rochkind, M., *Advanced UNIX Programming*, Prentice-Hall, 1985. (265 pp.) ISBN 0-13-011800-1

One of the better books available on UNIX system programming. Extensive example code for all the system calls available under AT&T System V UNIX. Some coverage of system calls under BSD UNIX and Xenix. Very readable.

*Stevens, W.R., *Advanced Programming in the UNIX Environment*, Addison-Wesley, 1992. (744 pp.) ISBN 0-201-56317-7

Excellent book on programming under UNIX in general. Covers both BSD (4.3) and System V UNIX. Lots of sample source code (the concept is learning by reading code), Covers all aspects of programming under UNIX, like I/O, file and directory handling, process control, signals, inter-process communication, daemons, database programming, printer, modem and pseudo-terminal interfaces. Lots of exercises. The sample code in the text is available via anonymous FTP through UUNET.

X Window System

The books listed below (except for Mansfield) are only recommended for X Window system programmers. Most users of the X (be it OpenLook or Motif) do not need to program the system and will find the user-oriented book by Mansfield to be most useful.

Cutler, E., D. Gilly, and T. O'Reilly, *The X Window System in a Nutshel*l, O'Reilly & Associates Inc., 1992. (424 pp.), ISBN 1-56592-017-1.

This manual forms a quick reference to the essentials of volumes 1, 2, 3, and 4 of *The Definitive Guides to the X Window System* (see below), providing faster access to the important information (Xlib and X toolkit functions, structures, data types, font conventions etc.) than the full programming and reference manuals (which may still be needed for detailed references).

Ferguson, P., and D. Brennan, *Motif Reference Manual* (volume 6B of *The Definitive Guides to the X Window System*), O'Reilly & Associates Inc., 1993. (920 pp.), ISBN 0-56592-038-4.

A good, complete reference manual for the Motif toolkit. Complements the *Motif Programming Manual* by D. Heller, P. Ferguson, and D. Brennan.

Flanagan, D., *Motif Tools, Streamlined GUI Design and Programming with the Xmt Library* (volume 6C of *The Definitive Guides to the X Window System*), O'Reilly & Associates Inc., 1994. (1024 pp.), ISBN 0-56592-044-9.

A book that is supposed to speed up Motif programming. It is a practical guide to Motif programming, with lots of tips and tidbits from other Motif programmers. See also the *Motif Programming Manual* by D. Heller, P. Ferguson, and D. Brennan.

Flanagan, D., *X Toolkit Intrinsics Reference Manual*, (volume 5 of *The Definitive Guides to the X Window System*), O'Reilly & Associates Inc., 1992. (916 pp.), ISBN 1-56592-007-4

The official reference book on X Toolkit intrinsics. Probably a must for people that want to do extensive X programming. This is merely a reference manual, not a textbook, and does not provide extensive examples (it goes together with the *X Toolkit Intrinsics Programming Manual*).

Heller, D., P. Ferguson, and D. Brennan, *Motif Programming Manual* (volume 6A of *The Definitive Guides to the X Window System*), O'Reilly & Associates Inc., 1994. (1016 pp.), ISBN 0-56592-016-3.

A good, complete reference book on programming for the Motif GUI. Some workstations (like SGI) may come with Motif manuals of their own, in which case this book is not required, and it then depends on the contents of these manuals, whether any of the other X programming or reference manuals are still required. With this book certainly a programmer would still need the manual on the X toolkit intrinsics (volumes 4 and 5 of that series) for basic window programming, as well as the Xlib manuals (volumes 1 and 2) for graphics and low-level programming.

Mansfield, N., *The X Window System: A User's Guide*, Addison-Wesley 1991. (344 pp.) ISBN 0-201-56344-4

A tutorial introduction to X that assumes no knowledge of X or any other windowing system. Describes the background of the X Window system, explains how to use the system, and provides information on ways to customize it.

Nye, A., *Xlib Programming Manual* (volume 1 of *The Definitive Guides to the X Window System*), O'Reilly & Associates Inc., 1992. (824 pp.), ISBN 1-56592-002-3

The official programming manual on lower-level X programming (Xlib); required for people who want to program graphics under the X Window system.

Nye, A., *Xlib Reference Manual* (volume 2 of *The Definitive Guides to the X Window System*), O'Reilly & Associates Inc., 1992. (1138 pp.), ISBN 1-56592-006-6

The official reference manual on lower-level X programming (Xlib). Required for people that want to program graphics under the X Window system. It goes together with the *Xlib Programming Manual*.

Nye, A., and Tim O'Reilly, *X Toolkit Intrinsics Programming Manual (Motif Edition)*, (volume 4 of *The Definitive Guides to the X Window System*), O'Reilly & Associates Inc., 1992. (674 pp.), ISBN 1-56592-013-9

The official programming textbook for programmers using the X Toolkit Intrinsics (anybody programming new X programs). It addresses both OpenLook and Motif, and the *X Toolkit Intrinsics Reference Manual* is more or less a mandatory addition and complement to that book. For lower-level graphics programming, volumes 1 and 2 most likely are also required, and higher-level programming for OSF/Motif is addressed in volume 6 of that series. (Volume 7 covers XView programming on Sun workstations, but because all manufacturers seem to be converging towards the Motif GUI, this can no longer be recommended.)

Smith, J. D., *X, A Guide for Users*, Prentice Hall, 1994 (350 pp.), ISBN 0-13-123795-0

A tutorial approach to the window manager as well as more advanced features of X. Provides extensive coverage of the X customization process.

UNIX System Laboratories, *OPEN LOOK Graphical User Interface—Programmer's Guide*, Prentice Hall, 1992. (ca. 700 pp.), ISBN 0-13-726605-7

A relatively readable book on programming for the OpenLook GUI environment (about as readable as a single book on that subject can be). Written for advanced C programmers. Covers OLIT (the OpenLook Intrinsics Toolkit). Written for generic System V UNIX systems, some of the procedures (compilation) described in this book will require adaptation for an implementation on Sun workstations. The examples are not bug-free; you will learn much by debugging as well.

Glossary of UNIX, Sun, and VNMR Terms

This glossary was written specifically for users of Varian spectrometers using Sun workstations and VNMR software. It is not meant to be valid for all computer users. Therefore, you will find different definitions for the same keywords in other computing environments. This material is also not meant to be read through or even to be learned from start to end, but rather to be a reference for help in reading Varian and Sun publications, and to be a means to facilitate communication between different VNMR users by creating a common base terminology. The glossary does not cover multiprocessor workstations and terms relating to server technology.

100baseT: Mechanical and electrical standard for 100-Mbps twisted pair fast Ethernet; *see also* Ethernet, twisted pair, RJ45.

10/100baseT: Newer Ethernet interfaces are handling both 10baseT (10 Mbps) and 100baseT (100 Mbps), with automatic switching between the two speeds. *See* 100baseT, 10baseT, RJ45.

10baseT: Mechanical and electrical standard for 10 Mbps twisted pair Ethernet. *See also* Ethernet, twisted pair, RJ45.

16-bit computer: Not well defined; for example, a CPU chip such as the MC 68000 from Motorola has a 24-bit address path (16-MB address space), a 16-bit data and instruction I/O path, and can perform 16- and 32-bit calculations. However, because the I/O bandwidth is usually the main speed bottleneck, such a CPU is often called a “16-bit CPU” (but often also 16/32- or 32-bit). Examples of other 16-bit CPU chips are the Intel 8086 and 80286 microprocessors, and the 6800 CPU from Motorola. *See also* 32-bit computer, 64-bit computer.

32-bit computer: Defines a CPU chip with 32-bit wide data and instruction I/O path. 32-bit CPUs are also laid out mainly for 32-bit calculations, and they typically have a 32-bit address path (4 GB address space). Examples for 32-bit CPU chips are the SPARC, MicroSPARC, HyperSPARC, and SuperSPARC CPU chips from Sun; the MIPS R3000; the PowerPC 601, 603 and 604 microprocessors from Motorola—all in the area of RISC computing. Examples of 32-bit CISC processors are the 68020, 68030, and 68040 CPUs from Motorola, and the Intel 386, 486, and Pentium microprocessors. Most current graphics workstations use 32-bit microprocessors in their CPU. *See also* 16-bit computer, 64-bit computer, RISC, CISC, CPU.

3D-RAM: New RAM technology jointly developed by Sun and Mitsubishi for the Sun Creator and Creator3D graphics controllers. Combines DRAM, pixel ALU (arithmetic logical unit), SRAM, and a VRAM video buffer on a single chip. Offers dramatically improved graphics performance over conventional VRAM. *See also* UltraSPARC, Creator, Creator3D, ALU, DRAM, SRAM, VRAM.

4.2BSD (4.2 version, Berkeley software distribution): The basis for the Sun operating system until SunOS 4.1.4 (Solaris 1.1). 4.2BSD was developed at UC Berkeley and derived indirectly from UNIX Version 7 (written at Bell Labs for the PDP-11 microcomputer).

64-bit computer: Defines a CPU chip with 64-bit data and instruction I/O path. 64-bit CPUs are laid out for 32-bit and 64-bit calculations, and they typically have a address path larger than 32 bits (over 4 GB address space). Typical examples for 64-bit CPU chips are the UltraSPARC CPU from Sun, the MIPS R4000, R4400, R4600, R5000, R8000, R10000, and the Motorola PowerPC 620 and 750 (also called G3). Graphics workstations with 64-bit CPUs are now becoming available. *See also* 16-bit computer, 32-bit computer, UltraSPARC, Ultra, CPU.

68000: *See* MC68000.

68020: *See* MC68020.

68030: *See* MC68030.

68040: *See* MC68040.

68881, 68882: *See* MC68881, MC68882.

abort: Irreversible interruption of a program by some external signal. Under UNIX, programs may be aborted by pressing keys such as Ctrl–c and Stop–a, by special commands like `kill`, or by aborting the parent process, such as exiting Suntools with active windows.

access: *See* permissions.

access time: Sometimes used for the average seek time to access tracks on disk drives (usually 8 to 15 ms), but may also characterize the speed at which RAM or ROM memory can be accessed (currently around 60 nsec on Sun computers). For disk drives, the access to individual sectors is longer than specified by the average seek time, for example, due to rotational delays. *See* latency.

acoustic coupler: Simple, low-priced modem that operates up to 1200 baud and does not require special authorization or installations on the phone system.

Acrobat Reader: Adobe software for viewing and searching documents in PDF (Portable Document Format) files. For the VNMR online manuals, Varian has switched from FrameViewer (and manuals in native FrameMaker format) to Adobe Acrobat Reader and PDF files. *See* PDF, Adobe, FrameViewer, online manual.

active window: Unlike standard Macintosh and similar PC-based software, all windows on a Sun are active processes, even icons. But only one window, called the active window, can have the focus for keyboard entries (although obviously the terminology is wrong). The window focus is indicated by a solid frame. The Sun windowing software (SunView) allows for two options: activate windows by clicking on them with the mouse or move the mouse pointer onto the window (difficult when using small windows). The focus can also be on iconized windows, which are then unfortunately not visualized, and you can (inadvertently) type text into an icon.

address: A number or a sequence of characters enabling computers to find and identify some hardware or a piece of software.

Adobe Systems: Software company that markets leading products for the desktop publishing (DTP) market, such as Adobe Illustrator, Adobe Photoshop, and Adobe Acrobat. Adobe recently acquired Frame Technology and is now also marketing FrameMaker, one of the most powerful DTP products on the market. VNMR manuals are written using FrameMaker. *See* DTP, PDF, Acrobat Reader, FrameViewer, online manual.

AIX: UNIX operating system on IBM workstations, such as the RS/6000 series. The current versions of AIX are 3.2 (POWER architecture only), and 4.1.4and 4.2 (these support both the POWER and PowerPC CPU architectures). Most AIX 3.2 applications should be compatible with AIX 4.1.4 and 4.2. So far, VnmrI runs under AIX 3.2 only. *See* IBM, UNIX, VnmrI.

alias: Alternate name for a command.

ALU (Arithmetic Logical Unit): Execution unit inside a CPU chip that handles integer calculations and logical decisions. Modern CPU chips contain multiple ALUs that can execute integer operations in parallel.

anonymous FTP: Special FTP account on public FTP servers that requires entering the user name “anonymous” at login. The account has no real password definition, but for logging purposes (FTP logins are logged into a file) the convention is to enter the user’s e-mail address as a password. Most Web browsers also have a convenient graphical interface for anonymous FTP sites. Anonymous FTP is usually used to download public domain software. *See also* FTP, Web browser.

ANSI C: Newer standard that was proposed for the C programming language. ANSI C differs in many ways from the “traditional”, “Kernighan & Ritchie” C (i.e., the one originally proposed by the authors of the language); the most prominent change is in the declaration of function arguments, which are defined within the parentheses, rather than after, as in K&R C. This provides for better argument type checking. ANSI C (but not K&R C) is a subset of C++. *See also* C, C++.

API (Application Programming Interface): A set of high-level library functions used in programming in or for certain software environments. Examples are the X Toolkit (used in X Window System programming), OLIT (OpenLook Intrinsics Toolkit), OpenGL, *See* OpenGL, OLIT, X Toolkit.

applet: Small Java program, compiled to byte codes, downloaded as part of a Web page, and executed on the Web client. Applets are used, for example, to produce an animated Web page or to allow for user interaction with the Web page without involving further communication with the Web server. *See* Java, HTML, WWW.

argument: Information passed to a command or subroutine (as part of the actual call) to further direct its operation.

ARPANET: The prototype, the predecessor, and later an early integral part of the Internet. The TCP/IP protocols were developed for ARPANET. *See* Internet, TCP/IP.

array processor: Specialized boards used on some systems (such as some Sun-3 systems) to speed up floating point operations on data arrays (vectors). Due to their architecture (using pipelines and simultaneous processing), array processors are very efficient on vectorized data, even using floating point operations, but they can not be used for scalar operations. The standard floating point hardware used on Sun workstations is equally well suited for vectorized and scalar operations, such that array processors are no longer needed. Array processors require modifications in the software, involve considerable processing overhead (like transferring the data into dedicated memory), and are available to only one process at a time.)

ASCII (American National Standard Code for Information Interchange): Represents characters with 7- or 8-bit numbers. ASCII is the most important standard for the transmission of text files.

ASIC (application-specific integrated circuit): Custom devices that help to simplify printed circuit boards by using fewer components. An example is the video chip used on newer Sun workstations.

assembly language: A low-level programming language that is very close to entering machine language directly. Allows writing very fast programs (normally faster than any compiler could do), but the complexity of these programs is limited, especially in large programs, which are hard to debug and modify because it is difficult to read and fully understand them. Assembly language is used mostly to speed up uniform, repetitive tasks on large data tables (such as Fourier transformations and pixel operations) or with non-standard data formats. The main disadvantage of assembly language programming is that it is intended for certain computer hardware; therefore, programs are usually not portable between different computers. Assembly language programming is usually limited to small modules called from within a high-level program.

asynchronous: Transmission in which the speed is determined by conventions such as the baud rate (in communication over a RS-232 cable) and not through synchronization signals (e.g., lines 15, 17, and 24 in a RS-232 cable). Asynchronous transmission is less demanding on the hardware side. Asynchronous transmission must always account for possible deviations from the convention (with RS-232, up to 38400 baud can be obtained in asynchronous transmission). Note that even with asynchronous transmission, the devices that communicate have some control over each other through handshaking signals that, for instance, enable the receiving device to stop the signal flow when its buffer is full. *See* handshaking.

AT&T Bell Labs: *See* System V.

ATM (Asynchronous Transfer Mode): New, emerging standard for networking, using either unshielded twisted pair (UTP) or fiber-optical transmission media (depending on the speed). ATM transmits all data in the form of 53-byte packets, which simplifies the ATM adaptor hardware and allows for potentially higher throughput than Ethernet. But unlike Ethernet, ATM transmission rates are isochronous (the transmission speed for a given link is kept constant for the duration of the connection) and not dependent on the network load (on Ethernet, increased collision rates at high loads cause a slowdown in the network efficiency). This makes ATM ideal for the transmission of voice and video data, such as in multimedia applications and video conferencing. Standard ATM speeds are 25, 155, 622, and 2488 Mbps. Sun currently offers ATM adaptors for 155 Mbps (over unshielded twisted pair) and 622 Mbps (over fiber-optic cable). Ethernet operates at 10 Mbps (standard Ethernet) or 100 Mbps (fast Ethernet, available on Sun as a SBus card or in Ultra workstations with Creator/Creator3D graphics). *See also* Ethernet, Ultra, UTP.

AUI (attachment universal interface) connector: Older Ethernet connector and cabling, with shielded 15-pin cabling and slide-lock connectors. AUI is standard for connecting transceivers for coaxial Ethernet (N-Type/Inline, Vampire/Thick Ethernet, and BNC/Thinnet) networks. Older workstations had an AUI connector that could be used to attach a transceiver directly or via AUI transceiver cable. Newer systems offer an MII connector that also permits connecting to UTB and fiber-optic networks (plus an RJ45 connector for 10/100BaseT), and for AUI transceivers an MII-to-AUI cable must be used. *See* MII, Ethernet.

authoring tool: *See* Web authoring tool.

background process: A process detached from the calling environment. The calling window remains accessible during the execution of the background process. Processes that produce lots of output are not suited for background operation, because the output may interfere with other programs running in the calling shell.

backplane: Hardware in the back of a card cage that links all boards through single- or multiple-bus structures. Every board fits into connectors in the backplane.

backup: To copy files to secondary storage (usually a tape or a optical disk on newer systems) for safety in case the originals are lost or unusable.

backup tape: Magnetic tape that contains a copy (duplicate) of data on a hard disk. Used for safety reasons so that data can be recovered if data becomes corrupt or if the disk dies due to a head crash.

bad track: Track on a disk where data cannot be written or read reliably. Most disks have bad tracks. During the surface analysis, which is done after formatting the disk, such tracks are labeled and subsequently replaced by other tracks. Bad tracks can eventually show up due to aging. In such a case, the disk has to be reformatted with surface analysis. Some disk drives automatically recognize bad blocks at runtime and assign spare blocks as replacement.

bandwidth: Characterizes the speed of a display (MHz) or a bus (MB per second). For screens, the bandwidth does *not* describe the number of pixels drawn per second (lines × columns × refresh rate ÷ interlacing, e.g., 1024 × 1280 × 66/1 = 86,507,520), but rather the speed at which the electron beam can switch between bright and dark (in a good monitor, bandwidth must be more than the number of pixels drawn per second).

batch: Confined (well-defined) amount of data.

batch mode: For mainframe computing, the execution of programs in packets (batches), where a program can be executed in a single batch or in several batches, not necessarily one after each other. Because the Sun is a multiuser system, many users could basically use the same plotter; therefore, the plots have to be plotted in batches (usually a full page) such that plot requests from different users are not mixed up on the same page.

battery: An in-chip battery keeps the real-time clock on Sun CPU boards running often for over 10 years.

baud rate: Speed of transmission in serial interfaces (modulations per second, corresponding roughly to bps, or bits per second). Some standard speeds for the RS–232 are 300, 600, 1200, 2400, 4800, 9600, 19200, and 38400 baud.

Berkeley UNIX: *See* 4.2BSD.

beta test: After extensive testing in the factory (software development group, applications laboratory, etc.), selected users test the beta version of software before it is released to customers.

bin: Name often used for a directory that contains executable software.

binary file: File containing information that can directly be interpreted by the CPU. Binary files include 16- or 32-bit integers, 32-bit floating point numbers or compiled programs (object code). Other file types are directories and text files.

bit: Single yes or no (1 or 0) information that makes up the smallest binary data unit.

block: A group of data on a disk or a tape that is transferred as a unit. With 4.2BSD UNIX, data are normally stored in 8-KB blocks. *See also* I–node.

BNC cable: A type of coaxial cable with 50-ohm impedance, used for screens, Ethernet, and other rf connections.

board: Hardware that can contain a complete computer or parts of it.

bootup: The process of starting an operating system. On UNIX systems, this involves a series of steps: loading the primary and secondary bootstrap, loading the kernel, finding all the available hardware, mounting file systems, checking file systems, starting up all the necessary daemons, and initiating the login shell in multiuser mode.

bootup messages: Messages automatically displayed after or during the bootup.

Bourne shell: Widely used UNIX command interpreter, common to all UNIX systems. On UNIX systems, Bourne shells are seldom used for interactive work but rather for shell scripts, where they are most efficient. *See* shell, C shell.

browser: *See* Web browser.

BSD (Berkeley Software Distribution): S*ee* 4.2BSD.

buffer: Memory for temporary data storage. A buffer is often used to link two devices with different inherent speeds, such as a computer and a printer.

bug: Error in software. Totally bug-free software for a large program is considered impossible. Software programmers can only try to reduce the number of bugs, a time-consuming process. *See* debugging.

bus: Hardware for transporting data between different devices. The receiving device can be addressed by the device that wants to send data, so that the data traffic occurs only between two specific units.

bus arbiter: Handles requests for data transfers and decides which device is allowed to become the bus master. *Compare with* bus master.

bus master: Only one device, the bus master, on a bus is allowed to control data transmission at a time.

bus terminator: Most buses require terminators for proper operation. For example, early SCSI bus devices had to be terminated by a termination plug or by a small chip (termination resistors) on the last device. Therefore, when changing the sequence of devices on a SCSI cable, users needed to check whether the bus was still properly terminated or else the bus and the entire system could be nonfunctional. Many modern SCSI devices are self-terminating: termination plugs are no longer required.

button: Switch that is activated by pressing, for example, the Sun mouse has three buttons. In a graphics user interface (GUI), menus can contain items called buttons that can be selected with a mouse click or by pressing a function key on a keyboard.

byte: A unit of 8 bits, commonly used to characterize the size of files, memory, or other storage devices, although the data might be organized in 16- or 32-bit words, or even 64-bit units or larger for floating point numbers.

C: A powerful, flexible programming language developed for UNIX by Dennis Ritchie based on Ken Thompson’s language B in the in the early 1970s. Although Pascal is usually considered better for learning and teaching, C produces very efficient (small and fast) object code. C comes in two “flavors”: the original, “Kernighan & Ritchie” (K&R) C, and ANSI C. K&R C was used to write most of the VNMR software. *See also* ANSI C.

C++: Superset of the C (actually: ANSI C) programming language, with tools and utilities for object-oriented programming (OOPS). Most implementations of C++ consist of a preprocessor that first translates C++ into C, followed by normal C compilation. SunSoft’s C++ uses a direct compilation technique for faster compiling. Different from C, C++ uses extensive internal argument and type checking, which makes it considerably safer than C. C++ comes with several powerful programming options, such as operator and function overloading, allowing the programmer to redefine the meaning of standard math operators, such that they work with complex objects. This feature makes it easy to define functions with variable number and type of arguments. Compared to C, C++ allows for new levels of simplification and abstraction in programming. Some VNMR utilities, such as the ImageBrowser, are written in C++. *See also* ANSI C, C, OOPS.

C shell: UNIX command interpreter developed and optimized for interactive work. Originally, it was specific to 4.2BSD UNIX systems, but now is mostly used for interactive work and writing shell scripts.

cache memory: Fast memory buffer that communicates directly with the CPU, thus avoiding unnecessary wait-states due to slower transfers to disks or ordinary memory. The most recent generation of workstations uses two levels of cache memory: 6 to 36 KB of primary cache on the CPU chip, and 0 to 2 MB external secondary cache on a separate chip.

capacity: Size of disks or memory, usually in kilobytes (KB), megabytes (MB), or gigabytes (GB).

caddy: Cartridge that holds and protects the CD-ROM in certain types of CD-ROM drives. Without caddy, a CD-ROM could not be inserted in these drives.

card cage: Structure that contains the backplane and the support for printed circuit boards.

case-sensitive: Pertains to systems or programs when the case of letters, either upper or lower, makes a difference. UNIX is case-sensitive in most matters, and the C programming language is always case-sensitive.

CDE (Common Desktop Environment): In response to Microsoft Windows 95, the major UNIX manufacturers (including Sun, IBM, HP, and Novell) decided to define a global standard for the X Window System and UNIX GUI. Sun's CDE is one of the first GUIs that comply with this new standard. CDE is mainly based on Motif standards. Sun brought in some of its desktop and development tools and some internal protocol definitions from OpenLook, while it had to give up many basic OpenLook features such as the look-and-feel. CDE has become the new standard Window interface with Sun and Solaris; it is included with Solaris 2.5. As of release 5.2F, VNMR is compatible with CDE (while maintaining compatibility with OpenLook). *See also* GUI, OpenLook.

CD-R (recordable compact disk): Optical disk (same form factor as used for audio disks) holding up to 644 MB of software or data. CD-R disks can be written only once and can serve as a permanent data storage for archiving data. Unfortunately, only a few manufacturers are building CD-R drives for UNIX workstations. Writing CD-R disks might be tricky because they typically need to be written once and in one continuous step; an interruption in the data flow might make the disk unusable. In the future, the DVD-R might replace CD-R drives. *See* CD-ROM, CD-RW, DVD-R.

CD-ROM (compact disk-ROM): Read-only optical disk (same as used for audio disks) containing up to 644 MB of software or data. Different from audio records, the information on CD-ROMs is organized in concentric, circular tracks (not in a single spiral track), and a CD-ROM is written starting with the innermost track. The Solaris 2.X operating systems and many software packages, such as VNMR, are exclusively available on CD-ROM. Sun UNIX documentation is also available on CD-ROM. The Sun CD-ROM drive can also be used to play audio CDs.

CD-RW (rewritable compact disk): Removable optical disk that can be rewritten just like a hard disk (same form factor as used for audio disks) holding up to 644 MB of software or data. Only a few manufacturers are building CD-RW drives for UNIX workstations. With the increased storage requirements, the DVD-RAM might replace the CD-RW drives. *See* CD-ROM, CD-R, DVD-RAM.

Centronics port: 8-bit parallel port for printing. A Centronics port is available on many printers, such as the LaserJet series of printers, and is standard on the SPARCstation and Ultra workstations. SBus cards with a parallel port are available for systems that don't have a built-in parallel port. Printing over a parallel port can be more than five times faster than printing through a serial port using RS-232.

CGI (common gateway interface): Defines how a Web client displaying a Web form (an interactive Web page) and a Web server communicate and interact. A user who submits a Web form causes a special character string (as defined in the form) to be sent to the Web server, to a specified processing tool, which is often a script written in Perl but can be a simple shell script or a program written in C or any other language). This software then causes further actions to be performed, including: HTML code is sent back to the Web client, mail is sent to the specified address, a database is searched, and the search results are sent back to the client. CGI is the definition of the interface between the form (the Web client) and the Web server. See WWW, HTML, Perl.

child: A process that is started from another process is called a child of that calling process, as long as it runs in foreground. When the parent of the child process dies or is killed, the child also dies automatically. Error output is normally sent to the parent process. The only way of detaching a child from its parent process is to run the program in background. Output is then still sent to the calling shell as long as that process is alive. Afterwards, the output is lost. *See* process.

CISC (complex instruction set computer): A type of CPU chip. Typical CISC chips are the Motorola 68000 series and most other 16- and 32 bit CPU chips used in personal computers and in older workstations. *Compare with* RISC.

class: An “object type definition” in object-oriented programming style (OOPS). A class defines the properties (private and public elements, constructor, destructor, etc.) of an object. An object is an instantiation of a class. *See* OOPS, class.

clicking: Pressing a mouse button. Most often, this involves moving the mouse pointer onto a particular object shown on the screen and then pressing a button (and in most cases releasing it again).

client: Computer system that uses facilities (usually disk space) on another computer (server). For the X Window System (X.11), the terms client and server have a different meaning. *See* server, X Window System, X client, X server.

clock: Most computers have a built-in real-time clock powered by a battery that typically lasts for 10 years or more.

clock frequency: Basic frequency (typically 50 to 200 MHz) at which a processor operates. This is *not* the rate at which CPU commands are executed, because many commands require several clock cycles, especially on CISC computers. For example, on the CISC Sun 3/60, a 20-MHz 68020 CPU chip performs at 3 MIPS (6.7 clock cycles per average instruction). With RISC computers, the clock rate is much closer to the command execution rate. For example, on a Sun SPARCstation 2, a 40-MHz SPARC chip performs at 28.5 MIPS (1.4 clock cycles per average instruction). The SPARCstation 10 and 20 series and the Ultra workstations have a special RISC CPU architecture (*see* SuperSPARC, UltraSPARC) that allows performance of 2.5 instructions per clock cycle. Other CPU architectures execute over 1 instruction per clock cycle by internally doubling the clock frequency.

coaxial cable: A high-frequency rf transmission cable that is intended to avoid interference with other rf devices.

collision: Happens when two devices want to transmit data simultaneously over Ethernet. Both devices try again after random delays. The number of collisions increases with the workload on the Ethernet.

command: Stand-alone program (or a shell script, macro, or menu button) that is started by entering its name or clicking on its menu button See *argument*.

command interpreter: Program that reads commands (such as operator input) and calls the necessary system functions to execute the command. VNMR has a built-in command interpreter, Sun UNIX comes with command interpreters called shells, and many interactive programs have a command interpreter built in. *See* shell.

command line: A line of characters containing one or more commands. Can also refer to a particular position on the screen display, where a command line can appear. The C shell on 4.2BSD UNIX systems includes an online command line editor that allows modification and execution of the current or previous command lines.

command mode: Special mode of operation with some editors (such as `vi`) where keystrokes do not enter characters into the text, but are interpreted as editor commands.

compiler: Program that converts source code (programming instructions that the programmer can easily read, modify and understand) into machine language for the CPU (instructions that are almost impossible to decipher).

computer: General term for an electronic machine that can perform various arithmetic, logical, and flow-control instructions at a high rate of speed.

configuration: A particular assemblage of hardware or software components.

connector: A piece of hardware that links a board to the backplane or a cable.

console: The device on which the operator works, including a keyboard and text output devices such as a screen or a printer. Under UNIX, the console is also the shell (or the window) on which system messages are presented. Output and error messages from programs running in background are often redirected into the console window.

constructor: Definition of actions, such as variable initialization, that occur automatically upon creation, or instantiation, of an object in object-oriented programming. *See* OOPS, object.

converged UNIX: UNIX version that combines parts of the major branches of the UNIX family—4.2BSD from UC Berkeley and System V from AT&T Bell Laboratories. Solaris is a converged UNIX, based on System V.4, containing most features from 4.2BSD.

coprocessor: Extra chip required for functions (such as floating point operations) that are not implemented in certain CPU chip models. For example, early SPARC CPUs, the MIPS R2000 and R3000, as well as the MC68000, MC68020, and MC68030 and several other early 32-bit CPUs did not have floating point capabilities built in and required a coprocessor chip to perform these calculations efficiently. The result was still slower than performing the floating point calculations by an on-chip floating point unit, but much faster compared to emulating the calculations with integer math. *See* MC68881.

core: The main random-access memory in a computer. The expression originates from the 1960s, when such memory was manufactured from tiny ferrite cores strung on wires.

CPU (central processing unit): The control, arithmetic, and logical unit of a computer. CPU can refer to a complete motherboard in a computer or to just the central processor chip, for example, 68020, SPARC, SuperSPARC, UltraSPARC.

crash: Sudden interruption of a program or the operating system from a software bug.

Creator: Graphics accelerator used in Sun’s Ultra 2 and high-end Ultra 1 workstations, such as the Ultra 1/170E, as well as the Ultra 10, Ultra 30, and Ultra 60. The Creator graphics board comes with 5 MB of 3D-RAM, allowing for 24-bit color at a screen resolution of 1280 × 1024 pixels. Creator offers unprecedented levels of graphics performance (windowing, 2D graphics, imaging, 3D wireframe) at a competitive price. *See also* Creator3D, Ultra 1/170E, Ultra 2, Ultra 10, Ultra 30, Ultra 60, 3D-RAM.

Creator3D: Graphics accelerator used in Sun’s Ultra 2, high-end Ultra 1 workstations, such as the Ultra 1/170E, as well as the Ultra 10, Ultra 30, and Ultra 60. The Creator3D graphics board comes with 15 MB of 3D-RAM, allowing for 2 × 24-bit color with Z buffering at a screen resolution of 1280 × 1024 pixels. Beyond the performance of the Creator graphics board, the Creator3D offers excellent performance in 3D solids imaging (such as used in molecular modeling and related applications). *See also* Creator, Ultra 1/170E, Ultra 2, Ultra 10, Ultra 30, Ultra 60, 3D-RAM.

CRT display (cathode ray tube display): Provides visual output of information.

cursor: An indicator on the screen (a line or an arrow) that can be moved around on the screen for pointing to some displayed object. A cursor is moved by keystrokes (as in some text editors) or by some mechanical or optical device such as knobs or a mouse.

cylinder: Most current hard disks contain several disk platters with two heads each. The heads move synchronously and read and write a series of concentric circles called tracks (not spirals like on audio records). A circle that *all* heads can read or write in a single turn of the disk is called a cylinder.

cylinder groups: BSD UNIX and Solaris 2.x organize cylinders into cylinder groups to minimize the average access time. Whenever possible, files are kept within a single cylinder group, such that the disk heads don't have to do large seeks to retrieve or write the data.

daemon: Programs that normally work silently in background and maintain UNIX services. Daemons are periodically active, either upon request or in some constant time intervals. Examples of UNIX daemons are the programs `update` (periodic sync to update the disk), `cron` (performs actions at specified times), and `Acqproc` (acquisition process).

DAT (digital audio tape): DAT drives are available for Sun computers. Such tapes serve as high-capacity storage media. A single tape cassette can hold 2.4 GB of uncompressed data (typically 5 GB compressed) and the transfer data at about 500 KB per second. Sun's 14-GB Exabyte drive (based on 8-mm video technology) is more expensive, but has twice the storage capacity and transfers twice as fast. *See* Exabyte.

data: General term for any kind of digital information, but can refer specifically to the information (numbers, text) a program works on. *data* originates from Latin and is the *plural* of *datum*.

data block: *See* block, I-node.

debugging: Removal of software bugs or errors. The major step in debugging is testing the software. Testing never ends—all software users are also software testers. Each user can help improve the performance of software by reporting bugs. Good bug reporting is very important. Whenever a bug is found, it should be documented thoroughly, including all error messages and the circumstances under which it happened (what was the exact sequence of commands used? What programs were running simultaneously or before the bug was detected?). Any additional observations about a bug may be important. If possible, try to reproduce the bug. And report all of this as soon as possible!

default: Instead of not working if certain inputs are not supplied, a program can also use some internally defined standard input that makes it a useful command without any input or with only partial input, thus saving the user time in entering the command.

degaussing: Due to the large static magnetic field in an NMR system, the monitor housing can become magnetized after some time of operation, resulting in color distortions. The magnetization can be removed by pressing the degauss button on the back of the screen as long as the display oscillates. Many monitors, such as the 17-inch and 20-inch Trinitron color screens on Sun workstations, are degaussed automatically when switched on.

delete: Removal of a file or a file system. UNIX uses the `rm` command to delete files.

DesignJet: Hewlett-Packard inkjet plotters for large format paper (up to size E / A0, even larger on some models). The DesignJet replaces the DraftPro and DraftMaster large-format mechanical plotters from HP. All DeskJet models support HP/GL and HP/GL-2 plotting. Some also support PostScript plotting. *See also* HP/GL, PostScript.

DeskJet: High-quality automatic paper feed inkjet printer from Hewlett-Packard that can print at 300 to 600 dots per inch, the same resolution as many laser printers. and can emulate HP LaserJet printers.

deskside workstation: Workstation in which the CPU is usually placed on the floor and *not* on the desk.

desktop publishing: *See* DTP.

desktop storage pack: External module for SPARCstations, containing a disk (2.1 GB), a QIC, DAT or Exabyte tape unit, or a 644 MB CD-ROM drive.

desktop workstation: Workstation where the CPU and display unit usually placed on a desk. The CPU (sometimes including hard disks) normally is placed under the monitor and often has just the same footprint as the monitor or even less. Most current workstations are desktop computers.

destructor: Definition of actions that occur upon deletion of an object in object-oriented programming. *See* OOPS, object.

device driver: Software, built into the UNIX kernel, that allows communication with external devices such as the acquisition computer.

diagnostics: Special software for checking hardware or software. Most computer hardware on a Sun can be checked in the monitor level. For help on monitor commands, type `h` and press Return. SPARCstations have a new monitor mode that uses the ok prompt. Enter `help` for more information.

dialog box: Window on the screen that displays some information and then waits for user input.

directory file: A file containing other files. Unlike older systems, UNIX directories are just a listing of other files on the disk and do not actually contain files. All UNIX files, including directories, are attached to a higher-level parent directory. The top-level directory in UNIX (`/`) is then attached to itself because the parent directory of `/` is `/` itself. *See* text file, binary file.

disk: Non-volatile magnetic storage media for large amounts of data. Used for data in continuous use or currently worked on. For safety reasons (data might be erased or overwritten, or the disk drive might fail), make backup copies of important files.

disk drive: Individual disk unit. Current disk drives used on the Sun for Varian spectrometers are 3.5-inch hard disk drives with formatted storage capacities of 535 MB, 1.05, 2.1, or 4.2 GB (SCSI disk drives), or 4.3 GB (EIDE disk drives).

disk partition: Under UNIX, a hard disk can be partitioned into up to 8 slices, or partitions. The first disk is normally partitioned into a root partition (`c0t3d0s0`), `/usr` partition (`c0t3d0s6`), `/var` partition (`c0t3d0s4`), `/opt` partition (`c0t3d0s5`), and `/export/home` (`c0t3d0s7`). A second disk is usually partitioned into a single slice (`/data`, `c0t0d0s2`).

disk slice: New term for disk partitions. With the transition to Solaris 2.x, Sun started calling disk partitions “slices”. The term is actually misleading, as a slice is a series of consecutive cylinder groups, rather than being slice-shaped. *See* cylinder groups, disk partition.

disk space: The amount of data that can be put onto a disk. In many computer systems (such as the VXR-4000 system), disk size is specified before formatting, so a “140-MB disk” may hold only 120 MB of actual data. On other systems, such as the Sun, disk space is specified after formatting, so a 2.1-GB disk holds 2.1 GB of data. But not all of this space may be apparent to the user. The swap space, for example, is actually a hidden space on the disk. Thus, the apparent size of the system disk is reduced by the amount of swap space.Additionally, in order to maintain disk performance at high levels, UNIX will report partitions being “100% full” when only 90% of the disk space is filled.

display: Usually the CRT screen, but often refers to the displayed information and not the hardware unit.

DNS (Domain Name Service): Software component that translates full Internet host domain names (e.g., `lal750.al.nmr.varian.com`) into a numeric IP address (e.g., `132.190.42.62`). DNS goes to domain name server hosts, who know how to query Internet address from the network, to find out about IP addresses. It then builds up an internal translation table, so that it doesn't have to check the servers for every request.

dot files: Special files for the customization of the user interface. In UNIX, these files have names that start with a dot (e.g., `.cshrc`). In normal directory listings using the `ls` command, these file are considered hidden or invisible, so their names do not show up unless the `ls -a` or `ls -A` command options are used.

dot matrix printer: Printer that composes characters from a dot matrix, as opposed to typewriters and daisywheel printers that produce entire characters from a single impact. Laser printers and ink jet printers are dot matrix printers. Dot matrix printers are more flexible because they can usually print graphics and several fonts.

double-indirect block: *See* I-node.

dpi (dots per inch): A unit of measure for the resolution of dot matrix printers or the pixel density of a display.

DRAM (dynamic RAM): A type of RAM chip, available currently with up to 64 megabits per chip. Dynamic means that it requires periodic refreshing (read and write back) operations. DRAM chips are slower than SRAM chips but are less expensive. DRAMS are used in Sun computers for the RAM memory. *See also* RAM.

DTP (desktop publishing): The creation of complex documentations, such as the VNMR manuals (typically with text, illustrations, tables, index, etc.) on a desktop computer (a UNIX workstation, a Macintosh, or a PC). VNMR manuals are created mainly by using FrameMaker software. *See* FrameMaker.

DVD (digital versatile disk): New, upcoming standard for digital, optical disks. DVD disks have the same form factor as the CD-ROM, but can hold much more information (4 GB and up to 17 GB, depending on the format, compared to 644 MB for a CD-ROM), due to a much higher information density. DVD drives can also read CD-ROMs. Even though CD-ROMs will continue to exist for many years to come, DVD drives will probably replace CD-ROM drives. DVD disks exist in five different formats: DVD-R, DVD-ROM, DVD-RAM, DVD-Video, and DVD-Audio (for the consumer market, no standard has been fixed yet for DVD-Audio). With current drives, double-sided disks (DVD-R, DVD-RAM, some DVD-ROM disks) must be taken out of the drive and inverted by hand in order to access the other side. Current DVD drives have average access times of around 200 milliseconds, and data transfer rates of around 2.8 MBytes/second. *See* CD-ROM, DVD-R, DVD-ROM, DVD-RAM.

DVD-R (DVD-Recordable): DVD disk that can be recorded (written) once (basically a WORM drive), with a capacity of 4 GB per side. DVD-R drives are currently still too expensive for the typical workstation user. *See* CD-ROM, CD-R, DVD, DVD-ROM, DVD-RAM, WORM.

DVD-RAM: DVD disk that can be read and rewritten just like a hard disk, with a capacity of 2.3 GB per side. DVD-RAM disks require special drives that can also read DVD-ROM disks, but DVD-RAM disks cannot be read on DVD-ROM or DVD-R drives. *See* CD-ROM, CD-RW, DVD, DVD-R, DVD-RAM, MOD disk.

DVD-ROM: Read-only DVD disk, with a capacity of 4.7 GB (single-sided/single layer), 8.5 GB (single-sided/double layer), 9.4 GB (double sided/single layer), or up to 17 GB (double sided/double layer). DVD-ROM drives can read DVD-R, DVD-Video, and DVD-Audio disks, but not DVD-RAM disks. *See* CD-ROM, DVD, DVD-R, DVD-ROM.

EBus (external bus): Bus for peripherals. The EBus is used in the latest PCI-bus-based Ultra workstations and connects to the keyboard, mouse, serial ports, parallel port controller, floppy drive, audio I/O controller, NVRAM and real-time clock, as well as EIDE disks. *See* EIDE, NVRAM.

ECC memory (error checking and correction memory): A device providing self-correction of 1-bit-per-byte errors and detection of 2-bit-per-byte errors. Only available in high-end workstations, such as the Sun SPARCstation 10 and 20, and the Ultra. *See also* SPARCstation 10, SPARCstation 20, Ultra.

editor: A program to modify text and data in files.

EEPROM (electrically erasable programmable ROM): Similar to EPROMs, except that EEPROMs can be erased by applying a specified voltage. Sun computers use EEPROMs to permanently store some configuration parameters that might be altered by the user. *See also* EPROM, ROM.

EIDE (enhanced IDE): Low-cost standard disk type often used in PCs. Ultra 5 and Ultra 10 workstations are equipped with 4.3-GB EIDE disks. *See also* Ultra 5, Ultra 10.

Elite3D: Graphics accelerator used in certain Sun Ultra workstations (Ultra 30, Ultra 60). The Elite3D graphics board provides top-end 3D graphics speed at screen resolutions up to HDTV format (the Elite3D internally performs floating point operations at 1.2 TFlops, i.e., 1.2×10^{12} floating point operations per second). Available with Ultra 10, Ultra 30, and Ultra 60 workstations. *See also* Creator, Ultra10, Ultra 30, Ultra 60.

e-mail (electronic mail): Exchange of information over a worldwide computer network that does not usually involve printing the message. Electronic mail is becoming extremely popular compared to telephone calls, faxes, or ordinary mail, because e-mail is fast (much faster than letters), it takes less effort than writing and sending a fax, and it does not interrupt the addressee’s work, but can be replied at the responder’s convenience. In addition to all that, digital information (text files, programs, etc.) can be sent from one computer directly to another computer. Typing and handwriting as a source for new errors can be eliminated. The Varian applications laboratories, technical support, and customer service functions can be reached via electronic mail.

emulation: Software simulation of a different hardware and software configuration. For example, standards such as DEC VT-100 have been created for terminals. Many other terminals now have the ability to emulate VT-100 and other standards. The same is true for printers and plotters. Some current standards are HP/GL for plotters and the HP mode, Epson, HP LaserJet, and Postscript standards for printers.

EPROM (erasable programmable ROM): Memory chip that can be erased by exposure to ultraviolet light. The chip can then be programmed again, using a PROM programming device. *See also* EEPROM, ROM.

ergonomics: Working with comfort at a computer for extended periods. Besides a comfortable chair at the correct height of the desk, the positioning of the screen and keyboard is important. To avoid eyestrain from screen reflections, a window or bright objects should not be located behind the operator.

error message: Message that informs the user about software problems or command entry syntax errors. Under UNIX such messages tend to be somewhat cryptic. You may need to ask a UNIX guru what some messages mean.

Ethernet: Xerox trademark for a type of hardware used to communicate between computers. The modulation of a 10-MHz rf signal transmits data, using special (complex) protocols (characterized as CSMA/CD, Carrier Sensing Multiple Access / Collision Detection). Ethernet uses special coaxial cable (up to about 2.5 km) or standard 50-ohm BNC cable connections (also known as thin Ethernet, only up to 500 m). For transmission, the cable must be terminated with a 50-ohm impedance load on both ends and branching is not allowed. A faster version of Ethernet, called fast Ethernet, operating at 100 Mbps over unshielded twisted pair (UTP, 100baseT) is now becoming available, allowing for much higher network throughput. *See also* fast Ethernet, 10baseT, 100baseT, UTP, Ultra.

Ethernet address: Every device on an Ethernet must have a unique address that consists of a series of hexadecimal characters of the form xx:xx:xx:xx:xx:xx. This address is hard-coded on a PROM into every Ethernet board. In Varian limNET software, this hardware address of the Ethernet board is used. In most modern Internet communication protocols, such as TCP/IP and NFS, a user-assignable software address is used, independent of the hardware address.

Ethernet terminator: Load with 50-ohm impedance on both ends of the Ethernet cable.

Exabyte: Company that manufactures 14-GB tape drives based in 8-mm video technology. The name Exabyte often is used as synonym for the tape drive itself. Such drives are available for Sun workstations. A single tape can hold 14 GB with hardware data compression, and the tape writes and reads data at about 1 MB per second, compared to the 75 to 90 KB per second of a 150-MB 1/4-inch cartridge (QIC) tape drive. Media costs are about 100 times lower per MB than with QIC drives.

executable: A compiled program or a shell script that can be executed by typing its name and pressing Return. The calling user must have execute permission. If desired, other users or groups can be restricted from executing the program or shell script.

exit: Leaving or quitting a computing environment such as VNMR, a shell, or OpenLook.

expansion board: Board added to the computer that enhances the capabilities of a system by adding more memory, more computing power (e.g., co-processors), additional I/O devices or capabilities (e.g., RS-232 expansion board), or higher graphics performance.

expansion memory: Memory module that expands the standard RAM memory. On current Sun workstations, the maximum total memory is between 256 MB and 1 GB.

expansion slot: Free space in a computer that allows insertion of an expansion board. All desktop Sun workstations have special expansion connectors for the SBus (one slot may be used for the color graphics controller). Larger workstations have extra expansion connectors for extra CPU modules.

fast Ethernet: Fast version of the Ethernet protocol that communicates between computers at 100 Mbps, compared to 10 Mbps for standard Ethernet. Fast Ethernet operates over unshielded twisted pair (UTP, 100baseT) and is now delivered with high-end graphics workstations such as the Sun Ultra 1/170E, as well as the Ultra 5, Ultra 10, Ultra 30, and Ultra 60. Alternatively, a fast Ethernet interface is available as a SBus expansion board. The Sun 100baseT Ethernet interface is autosensing. It does both 10 Mbps and 100 Mbps communication on the same network, adjusting itself to the speed of the system it is communicating with. *See also* Ethernet, 100baseT, UTP, Ultra 1/170E, Ultra 5, Ultra 10, Ultra 30, Ultra 60, SBus.

fiber-optic cable: Data transfers over computer networks at speeds above 200 Mbps cannot be achieved using electrical cables because of signal losses and rf interference, requiring extensive or expensive shielding. Instead, fast network protocols, such as 622 Mbps ATM, use fiber-optic cable (multimode optical fiber) for the data transmission. This has the additional advantage of providing for electrical insulation between networked computers, avoiding problems with ground loops. Unlike unshielded twisted pair cables, fiber-optic cabling avoids problems with rf interference. *See also* ATM, UTP.

file: For some programming languages, a file is defined as data that can only be read sequentially, with no random access, but now a file more commonly refers to any well-confined amount of data, with any structure.

file access mode: Sets the access protection of a UNIX file (who can read, write, or execute a file) among the three classes of users: the owner of the file, members of a specified group, or all others.

file system: Complex and well-confined system of a directory, its subdirectories, and all the files contained in the system. Under UNIX, a file system normally cannot be extended across disk partitions, although some features, such as the entire file systems mounted onto others, make the user believe that this is possible. Certain commands, such as recursive copying with `cp -r`, work only within local file systems, while others, such as `mv` on directories, work only within a single file system or disk partition.

filter: Program, or sequence of programs, that take input and modify the data for the output. For example, the filter `tr` replaces specified characters in the input with other character. Filter functions are often realized by using multiple pipes (each pipe indicated by a vertical bar), for example, `cat /etc/termcap | grep tvi | more`.

firewall: Software product, typically installed on a gateway computer, that shields a network from intruders. It makes the gateway transparent for users on the local network, while it blocks off most Internet protocols (such as FTP, telnet, rlogin, rsh, rcp, NFS, and HTTP) and only leaves access to simple protocols such as SMTP for e-mail from the other side. A firewall is often used to shield a company-internal network—local users may have full Internet access while outside users can only access people on the local net by e-mail. *See also* gateway.

firmware: Software permanently stored in ROM. On a Sun workstation, the firmware consists of the primary bootstrap and diagnostics software.

flickering: Visible oscillations on a CRT display (typically 1 to 20 Hz). Sun displays should be flicker-free. If this is not the case, look for transformers near the display, perhaps in a printer or other device.

floating point coprocessor: Dedicated processor chip for floating point operations. Early Sun workstations, such as the Sun-3 and early SPARCstation models, used a floating point coprocessor. Current Sun CPU chips, such as the MicroSPARC, the SuperSPARC, and the UltraSPARC, have this capability built in.

floating point number: Number in binary scientific notation, which has a binary exponent and mantissa. In computers, the exponent indicates powers of 2, not 10. This translates into the number of left- or right-shifts in the binary mantissa. Standard formats are 32-bit and 64-bit floating point numbers (sometimes 80- or 128-bit numbers).The 32-bit format uses an 8-bit exponent (–128 to 127) and a 24-bit mantissa, resulting in a range of $\pm 3\times 10^{38}$, the smallest numbers being in the order of $\pm 1.7\times 10^{-38}$, with about 7 significant digits. The 64-bit format (IEEE standard) uses 11 bits for the exponent (–1024 to 1023) and 53 bits for the mantissa, resulting in a range of $\pm\ 2\times 10^{308}$, the smallest numbers being in the order of 10^{-308}, with about 16 significant digits.

floating point operation: Calculation using numbers in binary scientific notation, which have a binary exponent and mantissa. *See also* floating point number.

floppy disk: Soft magnetic disk, usually either 5.25- or 3.5-inches in diameter. They are rather limited in speed and data size (up to 1.44 MB), but handy for transporting small amounts of data. A 3.5-inch floppy drive is optional with all Sun desktop workstations.

font: Character set with a specific graphic layout for every character. Dot matrix printers and sophisticated graphics terminals (such as the Sun display) can operate with a whole series of different fonts. Fonts can be of fixed width like most typewriters, where every character takes up the same space, or of proportional width, where the space allocated to each character depends on the natural width of the character.

footprint: Desk space occupied by a device such as a monitor or a CPU unit. Because many users prefer to place workstations directly on the desk, a small footprint has become important, although performance and the price-to-performance ratio are obviously most important. Sun computers combine a small footprint with good performance and price.

foreground process: Process called from some environment and remains attached to that process as a child of the calling process. Such a process is said to run in the foreground, relative to the calling process. Normally, any process can start only one foreground process at a time, and the calling process is then blocked until the foreground is finished or sent into background. Messages and output from foreground processes are sent to the calling process. *Compare with* background process.

formatting: Before a disk can be used, it must undergo a formatting process in which the disk is partitioned into cylinders, tracks, and sectors, and the information where every sector can be found is written onto the disk.

Fortran: The first high-level programming language. Its syntax looks somewhat archaic when compared to modern languages such as Pascal or C, but it still often used for scientific calculations and "number crunching" applications. For the Sun, Fortran is available as an option only.

FPU (floating point unit): Chip or integrated function of a microprocessor that performs floating point calculations. *See* CPU, microprocessor.

fragment: To avoid using too much disk space for small files, the Solaris disk file system can use fragments of 1, 2 or 4 KB for storing small files or remaining parts of large files, while large files are normally stored in 8-KB blocks. *See* block.

frame buffer: Dedicated memory for storing the pixel information for the display. Its size, in bytes, is calculated as *(number_of_pixels* × *bits_per_pixels* / 8). Sun normally uses 1 million pixels (900 × 1152) with black & white displays using 1 bit per pixel and color systems using 8 bits per pixel. Workstations with SX graphics, or those with graphics accelerators for 3D modeling (such as systems with Creator graphics) use 24 and more bits per pixel.

FrameMaker: One of the leading products in the DTP market, used predominantly for creating large and complex documents, such as the VNMR manuals. FrameMaker has been ported to UNIX, PC, and Macintosh systems. FrameMaker can export data into the PDF format. For some time, the VNMR online manuals were distributed in FrameMaker format, together with FrameViewer. *See* Frame Technology Corp., FrameViewer, DTP, PDF.

Frame Technology Corp.: Former software manufacturer of FrameMaker and FrameViewer. Frame has been acquired by Adobe, but their products continue to be available. *See* FrameMaker, FrameViewer, Adobe, DTP.

FrameViewer: Software for viewing but not editing FrameMaker documents. FrameViewer opened up a possibility to view FrameMaker documents without having to purchase a FrameMaker software license. For some time, the VNMR online manuals were distributed in FrameMaker format, together with FrameViewer. Recently, Varian NMR has switched over to Adobe Acrobat Reader and is now distributing the online manuals in PDF format. *See* FrameMaker, Acrobat Reader, PDF.

Free Software Foundation, Inc.: Organization "dedicated to eliminating restrictions on copying, redistribution, understanding and modification of software." Created in response to AT&T's licensing of UNIX software, the organization distributes widely used free software, such as gzip (GNU zip), GNU emacs, and GNU C/C++. This software is distributed as source code via public FTP servers such as `prep.ai.mit.edu` (for more information, read the file `/u2/emacs/GETTING.GNU.SOFTWARE` on that host). Because Solaris 2.x no longer has a C compiler included, Varian is using and distributing GNU C with VnmrX for Solaris. *See* GNU, GNU C/C++.

FTP (file transfer protocol): Protocol for transferring files via TCP/IP networks. The command `ftp` implements this protocol. FTP is typically used in interactive mode but only works if a password is defined for the remote account—unlike `rcp`, which only works if a password for the remote system is *not* required (`rcp` requires an insecure remote account or an entry in `/etc/hosts.equiv` or `~/.rhosts` on the remote machine). FTP is mostly used for transferring binary and text files and for downloading software and data over Internet. Public FTP servers usually use so-called anonymous FTP—a special FTP account that requires the user enter the name `anonymous`. There is no password definition for this account, but for logging purposes (FTP logins are logged into a file) the convention is to enter the user's e-mail address as a password. Most Web browsers have a convenient graphical interface for anonymous FTP. Some browsers also allow using FTP for downloading data from non-public FTP servers. These browsers prompt for user name and password, or alternatively these can be entered in the FTP URL. *See also* TCP/IP, anonymous FTP, Web browser, URL.

function: A subroutine in high-level programming languages. In C and related languages, functions have arguments for passing information that is specific to each function call, and functions return a value of a defined type to the calling environment.

function key: Keys (usually F1 to F12) that invoke special actions instead of entering characters. Often the functions assigned to those keys can be changed by the user. Most menu systems work in connection with function keys.

function overloading: In C, functions usually have a fixed number of arguments. Functions can be defined with a variable number of arguments, but this complicated and eliminates some security checking such as argument types. In C++, functions can be overloaded. Several functions can be defined with the same name, as long as each of the definitions, called prototypes, is different in the type or the number of arguments or both. The C++ compiler then matches the proper prototype with each of the function calls and provide full argument type checking. *See* C++.

GaAs (gallium arsenide): Potentially extremely fast chip technology. GaAs transistors switch much faster than silicon semiconductors, allowing for clock rates of several hundred MHz. This technology is not used now in workstations because current technology only allows for a rather low integration level (increasing the number of components), and expensive on-board peripherals, such as memory chips, would be required. *See also* multiprocessor system.

games: Programs that do not help promote science or increase spectrometer throughput. Sometimes, they might help an operator to make it through a long, boring acquisition.

gateway: Machine with network interfaces that link the attached networks. Because all communication between the attached networks must pass the gateway computer, gateways are frequent targets for hacker attacks. In industrial environments, gateways are often equipped with a firewall software that restricts certain Internet protocol levels such that they can only be initiated in one direction. *See also* firewall, hacker.

GIF (graphical interchange format): Standard file format for storage of bitmap files. Identified by a `.gif` extension, GIF files are widely used for images in HTML documents. GIF files can contain sequential display of images as well as interlaced images. In interlacing, images are first displayed with low vertical resolution that is gradually enhanced over several passes. This often permits seeing the contents of an image before the entire file is displayed. GIF is limited to 256 colors, is lossless, and has limited compression capabilities JPEG, an alternate to GIF, uses a more efficient compression mechanism, but is somewhat lossy (loses some data during compression). *See* HTML, JPEG.

gigabyte (GB): 1,073,741,824 bytes. 1 GB is equal to 1024 megabytes (MB) or 1,048,576 kilobytes (KB).

GNU (GNU is Not UNIX): Label used for Free Software Foundation software, which includes GNU C/C++, GNU emacs, and GNU zip (`gzip`). *See* Free Software Foundation, GNU C/C++.

GNU C/C++: C and C++ compilers from the Free Software Foundation. Because Solaris 2.x no longer includes a C compiler, Varian includes the Solaris GNU C for use with VnmrX. GNU C is used only for PSG software (`psggen` and `seqgen`) and for the compilation of other source facilities distributed with standard VNMR (such as `wtgen`). The VNMR Source Code Kit requires the SunSoft C compiler (SPARCompiler C or SPARCworks Professional C) and does not work with GNU C/C++. *See* Free Software Foundation, GNU, Solaris 2.x.

graphical user interface (GUI): User-interface software that enables using a computer by addressing (pointing to, clicking, dragging, etc.) graphical objects (icons, windows, scroll bars, buttons, check boxes, etc.) on the screen with a mouse or some other graphical pointing device, instead of using the keyboard for typing commands. The GUI was invented at Xerox for the Star computer and was then taken up by Apple for the Lisa and Macintosh computers, Microsoft for Windows operating system, OSF for Motif, Sun for SunView and OpenLook, and other companies.

graphics controller: Board (or on-board) function that contains the display frame buffer. This buffer is 1 MB on 8-bit color systems and over 3 MB on 24-bit color systems. On 8-bit color systems, this results in 256 possible numeric values per pixel. The graphics hardware can produce 16,777,216 different colors, $256 \times 256 \times 256$ = 8 bits or 256 values for each of the three colors red, green and blue. One of these 16 million combination is assigned by software to each of the 256 color numbers, and lookup table is used to translate the color numbers from the screen buffer into numeric values for the three DACs for red, green and blue. 24-bit color systems can produce all the 16,777,216 different colors simultaneously.

graphics workstation: High-performance stand-alone computer with high-resolution and fast graphics capabilities. Graphics workstations usually have multiuser capabilities but are equipped with a single high-resolution graphics terminal.

ground loop: Ideally, all grounds should originate at the same point, and ground loops should not be produced by having several grounding connections between two devices. Ground loops can lead to electrical oscillations, which can produce nasty interferences and random signal distortions that are difficult to cure. The simplest way to avoid ground loops is to have the shielding of all electrical cables only connected at one end (ideally to the central device, not to the peripherals) and to have one single, well-defined grounding connection between all devices.

GUI: *See* graphical user interface.

guru: Somebody who has worked with UNIX so much that the person can solve the most difficult problems from experience and may even understand UNIX error messages!

GX: Single chip 8-bit color graphics accelerator for fast windowing and 2D/3D wireframe graphics. This chip has been used in the SPARCstation 1, SPARCstation 2, SPARCstation IPX, and the SPARCstation LX, but more recently it has been substituted by the TGX (TurboGX) graphics accelerator that provides higher performance. For the latest, PCI-based Ultra workstation, a PCI-based GX graphics controller is available. *See* TGX, PGX.

hacker: Real computer freak who is in a permanent competition for the "best," most complex, most useless, and most cryptic program. This person would like to become a guru, but often ends up trying to prove to others that he or she is able to break into anybody's data files and software.

HAL (Host-to-Acquisition Link): Earlier Varian NMR spectrometers, such as the UNITY*plus*, UNITY, and VXR-S, used the SCSI bus to communicate with the acquisition computer. The HAL board formed the SCSI interface between the acquisition CPU and the Sun computer. This board had a Motorola 68000 CPU and 2 MB of memory. Current spectrometers (such as the UNITY*INOVA* and *MERCURY*-Series) use Ethernet instead of the HAL board to link the Sun and the acquisition CPU.

handshaking: Most interfaces require several lines for handshaking, which is a type of hardware or software protocol for data transmissions. For example, the receiving device tells the transmitter that its buffer is empty and that the transmission can start. With most bus traffic, the handshaking is a part of the bus definition, and the user should normally not be concerned about it. An exception is that in RS-232 communication, data flow is controlled by handshaking signals, but there are several handshaking options, unfortunately. Many RS-232 devices allow selection of different handshaking options. Often, unreliable data transfer via RS-232 occurs either due to improper setting of the handshaking options on one of the communicating devices or due to improper cabling of the handshaking lines. The simplest form of RS-232 handshaking is the back-transmission of special characters XON and XOFF to start and stop the data flow. More elaborate handshaking protocols involve dedicated lines. *See* null modem, XON/XOFF, RS-232.

hangup: Describes the status of a program that got trapped in a loop and can only be accessed by stopping the program. *See* interrupt, kill.

hard disk: General expression for rigid disks, such as SCSI hard disks and similar devices, as opposed to floppy disks that are commonly used on personal computers. *See* Winchester.

hard links: UNIX allows multiple entries in a directory to point to the same file. All these entries are equivalent and are called hard links. The file pointed to is only removed when all hard links are removed. The hard links to a single file can even be in different directories, but this is not a recommended practice because it destroys the tree structure of a file system. More than one hard link to a directory file is not permitted, because this would allow forming a loop in the file structure, which violates the tree structure and the hierarchical organization of directories. Hard links are not permitted between different partitions. *Compare with* symbolic links.

hardware: Basically, anything in a computer you can touch.

head crash: The heads writing onto or reading from hard disks fly only a few micrometers over the disk surface rotating at 5400 or 7200 rpm. A head crash occurs when any particle of dust, smoke, or fingerprint collides with the head. This can destroy the head and the disk so that all data on the disk are lost, and the disk has to be replaced. Today's hard disks are sealed; therefore, such events should occur very seldom.

help: Assistance for operators who do not know how to use a system. Help is available from manuals (hard copy and online) and from experienced users, such as a guru.

hierarchy: Hierarchical, parent-and-child structures are involved in UNIX software as well as in UNIX file systems. *See* hierarchical file system.

hierarchical file system: Storing files such that information is stored at different levels. The UNIX file system is hierarchical in that each location in the file system is called a directory and each directory is a collection of files and other directories, forming what is called a tree structure.

high-level language: A programming language, such as Fortran, Pascal, C, or C++, that takes an abstract view of programming a computer, in contrast to an assembly language that relates to certain computer systems only.

history: Stack register for the last command lines. Can be used to reexecute entire command lines without retyping them or to reconstruct the sequence of previous commands in case of problems with software.

home directory: The directory that becomes the working directory immediately after login. For normal users, it is also the directory where all or most personal files are located. The home directory is specified in the file `/etc/passwd`.

host: Computer that controls other computers. For example, on Varian spectrometers with Sun workstations, the acquisition CPU, a computer on its own, is controlled by the host, which is the Sun CPU. In UNIX, the term host is also used for any computer in a network.

host name: Each host computer is given a common but unique name, which makes easier for users to address it because they don’t have to type or learn a complex sequence of numbers or funny characters when talking to other computers.

host number: A number by which the software addresses and recognizes another CPU internally. Every CPU in a network must have a unique host number and name.

HotJava: Sun’s Web browser. HotJava is written entirely in Java and, of course, it fully supports Java applets. *See* Java, Web browser, HTML.

HP-GL (Hewlett-Packard Graphics Language): Internal computer terminology that defines the output on HP plotters. All HP plotters use HP-GL, and now the HP LaserJet III, 4 and 5 laser printers can interpret HP-GL/2, a terser version of HP/GL. HP-GL/2 speeds up data transfer times by a factor of over four compared to sending the entire dot matrix (about 1 MB per page at 300 dpi, or 4 MB at 600 dpi). *Compare with* Postscript, dot matrix printer, plotter.

HTML (HyperText Markup Language): A simple formatting language that permits formatting text in a Web browser, such as Netscape, HotJava, or Internet Explorer, by inserting simple ASCII formatting codes into the text. The codes can be added by any standard UNIX text editor. The formatting is *relative*. The HTML author selects styles such as heading levels, emphasized, strong emphasis, and relative font sizes, and some primitive paragraph styles, tables, list formats, etc. The actual formatting only occurs in the viewer’s browser. The font selection, width of the window, and many other formatting variables. are under the viewer’s control and can be adjusted to match the viewer’s screen size and viewing convenience. The real power of HTML lies in the use of hypertext links that permits incorporating any Web document that is accessible anywhere in the world into a local document. This document will have further hypertext links, building an enormous virtual library that can be reached and viewed through simple mouse clicks. HTML also permits incorporating pictures, sound, animated and interactive objects into a Web page. This can be a simple image (a GIF or JPEG file) or an animated audiovisual object defined using the Java programming language (which can be incorporated into HTML documents). *See also* WWW, Hypertext link, Web browser, URL, GIF, JPEG, Java.

HTTP (HyperText Transfer Protocol): To retrieve information or data from another machine on the Web, a Web browser needs to invoke a dedicated protocol. This protocol is specified in the URL of any remote hypertext link. For HTML documents from a Web server, the protocol is HTTP. On UNIX Web servers, HTTP is installed by starting special Web server software, such as `httpd` (the HTTP daemon). HTTP has become the dominating protocol of the total Internet traffic, overtaking the FTP and e-mail packet protocols. *See* HTML, URL, Hypertext link, Web server, FTP.

hub, Ethernet hub: Connection point for several 10BaseT or 10/100BaseT Ethernet connections, used to build star-shaped network topologies using twisted (STP or UTP) pair cabling. Hubs are typically equipped with 4 and up to 16 or more RJ-45 connectors; several hubs can also be interconnected for more complex network topologies. *See* 10BaseT, 10/100BaseT, Ethernet, RJ45, STP, UTP.

Hypertext: *See* HTML.

hypertext link: Any portion of an HTML document—including text and images—can be defined as a hypertext link for which an URL defines a different location in the same document, in a different document, or a document on any remote hypertext server that can be reached on the Internet. Clicking on the link from within a Web browser retrieves and displays the document (or the portion of a document) that the hypertext link points to. *See* HTML, URL, Web browser.

HyperSPARC: High-performance SPARC-based CPU chip architecture, designed by Ross Technology Inc., a subsidiary of Fujitsu, Ltd. A 150-MHz HyperSPARC CPU with 512-KB cache memory is used in the SPARCstation 20/151SX (sold by Varian for some time), which performs at 169 SPECint92 and 208 SPECfp92. This is the highest performance single-processor SPARCstation 20 from Sun. *See* SPARCstation 20/151.

I-node (index-node): UNIX directories consist of I-nodes (index blocks) and names. Under Solaris and 4.2BSD UNIX, I-nodes are 128 bytes long and the names can be any length. Information in the I-node includes permissions, owner and group, number of hard links, dates of creation and last modification, addresses of the first 12 data blocks (8 KB each), address to the first indirect block (with the addresses of the next 1024 data blocks), and the address to a double indirect block (with 1024 addresses of further indirect blocks). Therefore, the theoretical maximum size of a single data file in 4.2BSD UNIX is about 8 GB. For small files, the I-node can point to fragments (blocks of 1, 2 or 4 KB).

I/O (input/output): Input and output of data, mostly used for describing printing and plotting (for output), and mouse and keyboard entries (for input). Input is the transfer of information into a computer; output is the transfer of information out of a computer into a form that can be read or used by human beings or computer programs. UNIX and VNMR programs, as well as shell scripts and macros, can also take input from other programs.

IBM (International Business Machines, Inc.): Presently, the largest computer manufacturer worldwide (mainframes, minicomputers, workstations, and PCs); manufacturer of the RS/6000 series of graphics workstations, based on the IBM POWER and PowerPC families of microprocessors. VNMR is also available for IBM RS/6000 workstations, *see* VnmrI.

icon: Small picture or symbol on the screen that represents a window that is closed, or currently not shown. Unlike with most personal computer software, UNIX events can also happen inside windows. Programs, even graphics programs, can run and icons can be active windows, the window with the focus, and take keyboard input. The icon itself can act as a small graphics window, such as the clock tool, for example.

IDE (Integrated Device Electronics): Low-cost standard hard disk type that is most common in PCs. Ultra 5 and Ultra 10 workstations are equipped with 4.3 GB EIDE (enhanced IDE) disks. *See also* Ultra 5, Ultra 10, EIDE.

indirect block: *See* I-node.

ink jet printer: Dot matrix printer that generates dots by spraying minute droplets of liquid ink onto the paper. Obtainable resolution is up to 360 or more dpi (dots per inch).

insert mode: In text editing programs, a special mode in which the characters entered on the keyboard become text and are displayed on the screen. Some editors (such as the UNIX `textedit` program) operate exclusively in the insert mode (commands are available via menus). In `vi`, the most popular editor under UNIX, the insert mode has to be exited by pressing the Esc key in order to enter commands. *Compare with* command mode.

insertion point: In text editors, the point where text typed on the keyboard is inserted into the text, after going into the insert mode.

installation: Sun workstations are normally installed by qualified service personnel in order to avoid damage to the equipment. Should you plan to install it yourself, carefully follow the Sun and Varian installation manuals.

integer: On the Sun, an integer is normally a 32-bit integer number and can have values between –2,147,483,648 and 2,147,483,647. A short integer is 16 bits and can have values between –32,768 and 32,767.

integration: The integration level refers to the complexity of a single chip as well as to the number of chips and other devices on a single computer board.

interface: Device that links two devices, provides the necessary translation, and handles the required protocols on both sides.

interlacing: The cathode ray in the CRT display switches between the even and odd display lines instead of covering each line in succession. As a result, each pixel is refreshed with half the display refresh rate, thus reducing display hardware cost. To achieve an optimum, flicker-free display, Sun CRT displays do not use interlacing.

interleave factor (IL): Number that characterizes the distance of two sectors on a disk that can be read or written subsequently. IL = 1 means that all sectors can be read one after another (1, 2, 3, 4, ...). IL = 2 means that the sequence of sectors to be read or written on a disk with 17 sectors is 1, 3, 5, 7, 9, 11, 13, 15, 17, 2, 4, 6, 8, 10, 12, 14, 16, and so on, for higher interleave factors. The size of the interleave factor is dependent on the disk controller and on the type of disk used. Obviously, IL=1 is the fastest solution, because an entire track can be read in a single disk revolution (IL is also the number of turns that are required to read a complete track). The SCSI disk drives used in Sun workstations use IL = 1.

Internet: In 1969, the U.S. Defense Advanced Research Projects Agency (DARPA) funded a project for creating an experimental packet-switched network that would still be functional after losing some individual connections or network nodes. Different from a telephone network, which is connection-oriented, the new network would send information in short packets. The network software (protocols) would automatically split and recombine these packets, which are all individually and automatically routed over the network. A packet loss would cause automatic retransmission of the packets. In 1975, the network, ARPANET, became functional, but network research and protocol development continued further, and in 1983, TCP/IP were adopted as military standard. At that time, the term Internet started being used for combination of MILNET (the unclassified part of the defense data network, DDN) and the smaller, classified ARPANET. ARPANET was discontinued in 1990, but the Internet had in the meantime grown enormously. Internet now refers to the sum of the world-wide, interconnected networks that use the TCP/IP protocol family. Until recently, the vast majority of the packets travelling over the Internet were e-mail and FTP packets, but by 1996, HTTP packets have become the dominating portion of the overall traffic, due to the popularity of the World Wide Web. Sometimes, the term Internet is incorrectly used as synonymous with World Wide Web, which specifically refers to the part of the Internet that is accessed via Web browsers. *See* WWW, TCP/IP, FTP, HTTP.

interpreter: Program that reads source code and translates it directly into machine instructions for the CPU. There are also interpreters that read semi-compiled code such as

P-code. Interpreted programs always run slower than compiled programs. A special form of interpreter is the command interpreter such as the C shell and Bourne shell.

interrupt: Signal that interrupts the execution of software. Interrupt signals might be generated from the operating system or by the operator (by pressing some special keys). An interrupt might be pressing Ctrl-c to interrupt the execution of a particular program or pressing Stop-a to interrupt the CPU.

intranet: While the World Wide Web is used more and more to do business and share information with potential and actual customers, companies have realized that Web servers are also an excellent tool to share and distribute information internally, even a tool to access internal resources such as databases. Such facilities, called an intranet, are usually installed on internal networks that are not accessible to the public. Intranets are now rapidly gaining popularity all over the world. *See* WWW, Web server, Internet.

invisible file: File that is normally not shown in directory listings except if special commands or options are used, such as the dot-files in UNIX.

IP (Internet Protocol): Network protocol that is the basis of many higher level protocols, including TCP and UDP. The IP protocol provides a basic software connection at the information transfer layer. Error checking is done at a higher levels such as TCP or UDP. *See* TCP, UDP, TCP/IP.

IPC: *See* SPARCstation IPC.

IRIX: UNIX operating system on SGI workstations. VnmrSGI runs under IRIX 4.0.5, 5.x, and 6.x; the current version of IRIX is 6.2. *See* SGI, UNIX, VnmrSGI.

ISDN (Integrated Systems Data Network): A new standard for digital telephony that is rapidly being adopted and installed, especially in Europe. A basic ISDN connection offers a total bandwidth of 144 Kbps, split into two data channels of 64 Kbps each, and a control channel with 16 Kbps. There is also a primary connection option, offering 30 data channels with 64 Kbps each, almost 2 Mbps, and a control channel with 64 Kbps. An ISDN connection can be used for both digital and analog telephony. Direct digital PPP connections can be established at 64 or 128 Kbps via ISDN, provided the two computers are equipped with an ISDN interface. Such ISDN connections are faster and more reliable than standard modem connections. ISDN interfaces are available for Sun and UNIX workstations as well as PCs and Macintoshes. The ISDN interface is standard on the SPARCstation 10. For other Sun workstations it can be implemented with s a SBus expansion card. Using an ISDN port requires Solaris 2.x and a SunISDN software enabling kit. *See also* modem, PPP.

ISP (Internet Service Provider): Institutions and companies that provide Internet connectivity and operate the international Internet connections. Note that some ISPs only serve academic institutions (a part of the Internet is operated with public funding), while others operate in the commercial segment. *See* Internet.

Java: Object-oriented programming language, similar to C++, developed by Sun especially for Web applications. The main objectives in creating Java were compactness of the executable file, security, and portability. For security, dangerous language elements (i.e., elements that could be used to create viruses, such as pointers) from C++ were removed, and new security features were built in. Typically, Java is compiled into very compact byte codes that are platform-independent, sent over the Web, and then interpreted by the Web browser. The intent is to download not just HTML code over the Web, but small applications, called applets, that can animate Web pages. In the future, entire applications, including word processing and database software, might be kept on a central intranet Web server and distributed as applets that run on any platform. This dramatically simplifies the administration of desktop software. For performance-critical applications, there are also

Java native compilers that produce machine language executables, but that makes the application nonportable, of course. *See* OOPS, Sun, C++, applet, WWW, HTML, Intranet.

JavaScript: JavaScript was introduced by Sun and Netscape and complements the Java programming language. Java is compiled into byte codes and downloaded over the Web, whereas JavaScript code, which resembles Java source code, can be incorporated directly into the HTML code for a Web page. The JavaScript is then interpreted along with HTML by the Web browser. *See* Java, HTML, applet, Sun, Netscape, Web page.

Joy, William: Author of the C shell, a major contributor to Berkeley (BSD) UNIX, and a cofounder of Sun Microsystems.

JPEG (Joint Photographic Experts Group): Standard file format for storing bit-mapped graphics such as scanned images. JPEG format images are often used for images in HTML documents. JPEG is has no limitation in the number of colors and includes good compression but there may be some image quality losses due to JPEG compression, which is lossy. JPEG permits storing images with two-dimensional interlacing. Such images are first displayed with very little detail and are then gradually refined in several passes. *See* GIF, HTML.

jukebox: Device that contains a series of optical disks but only a single optical disk drive. It changes disks automatically and efficiently, such that huge amounts of data (up to several terabytes) are accessible within less than a minute.

jumper: Normally a small rectangular piece of plastic that sticks on two pins to create an electrical bridge used to configure computer boards for different environments. Jumpers are often also used to set the hardware address for devices, like disks and disk controllers, that are attached to a bus (e.g., the AP bus in a spectrometer or the SCSI bus).

kernel: The program block that always resides in memory, a central part of the UNIX operating system. The kernel on Sun workstations running Solaris 2.x is over 2 MB. Because the kernel is never swapped out, the amount of RAM available to application programs is reduced by the size of the kernel.

keyboard: Sun workstations come with an American standard (QWERTY) keyboard. Solaris supports an 8-bit character set; therefore, non-US keyboards can, in principle, be used on Sun workstations.

kill: Stopping the execution of a program. Under UNIX. the `kill` command allows stopping applications programs as well as daemons or any running process.

kilobyte (KB): Corresponds to 1024 bytes.

LAN (local-area network): A computer network in the same building or same site

LaserJet: Trademark for a laser printer from Hewlett Packard. Recent models are the LaserJet III, 4,and 5, which also can interpret HP-GL/2 and Postscript (with an optional cartridge). The LaserJet III operates at 300 dpi, and the LaserJet 4, 5 and 6 have a resolution of 600 dpi in both directions. *See* laser printer.

latency: On disk drives, the time to access a sector after the right track has been accessed. This time is influenced by the interleave factor. The rotational latency is inversely proportional to the speed of rotation, which is 5400 or 7200 rpm on current hard disks. *See* seek time, zero latency.

laser printer: Dot matrix printer that uses a pulsed laser beam to produce the individual dots that form the output. *See also* LaserJet.

Lexmark: Subsidiary of IBM that markets inkjet color printers. The current model, available with Varian spectrometers, is the 4079*plus*. This model comes with parallel, serial and Localtalk interfaces, which can be in combinations, such as from a workstation and a

MERCURY-Series spectrometer). The 4079*plus* accepts HP/GL and PostScript input and works with both A (A4) and B (A3) size paper. *See also* HP/GL, PostScript.

license manager: Many licensed software packages, such as FrameMaker, Sun's compiler products, and others, are not licensed on a per-installation basis, but rather for a certain number of users within a network. The main reason for this is that in many locations the software will be installed on a central server only, and several users will use the same software. In such situations, license checking is done via license manager software, a dedicated piece of software usually delivered with the licensed software package. When a user launches software that uses a license manager, this software first calls the license manager, which searches the network for other computers and users running the same software package. This process usually takes less than a minute, and then only the actual software starts. To eliminate this extra time every time a software package is needed, users can just close the window into an icon, but leave the software running for as long as they are logged into a system.

limNET: Trademark for Varian's Ethernet protocol for communication between VXR-4000, Gemini, XL systems, and computer systems running under UNIX or VMS, such as Sun computers, IBM RS/6000, or the VAX/MicroVAX series from DEC.

line conditioner: Device that reduces or eliminates surges and spikes in the ac power supplied to the system. A line conditioner is required where the mains power supply suffers from serious surges or high-voltage spikes.

link: *See* symbolic link, hard link.

linker: A program that links object files (precompiled program modules) to for an executable program. *See* object file.

login: Before being allowed to use UNIX and most other operating systems, the user must follow a login process by providing a user name and a password. If the user name and password are not typed in correctly, the multiuser environment cannot be entered. If the user name is not known to the system, it asks for a password anyway, so it is not obvious whether a user has not been defined or whether just the password was wrong. Only the superuser can create new users and reset forgotten passwords.

login shell: Program that handles the login (checking for user name and password), thus preventing outsiders from entering into the multiuser mode. *See* login.

logout: After finishing a working session on a multiuser system such as UNIX, a user should complete the logout process so that another person can only use the system when passed through the login shell. *See* login, login shell.

lunchbox: Nickname for Sun's desktop storage pack, the enclosure format for external disk and tape drives used with SPARCstation systems before the UniPack enclosure format was introduced. *See* desktop storage pack, unipack.

macro: VNMR term for a file that consists of a series of VNMR commands. A VNMR macro is called by entering the name of the macro file and is interpreted like a series of commands entered on the keyboard. Macros allow for a series of programming utilities such as branching, loops, and arguments, etc. They are useful for customizing and simplifying repetitive tasks. The same is possible under UNIX with shell scripts.

MAGICAL (MAGnetics Instrument Control and Analysis Language): Varian trademark for system software that supports full automation of spectrometer operation and analysis. MAGICAL enables the spectrometer to make decisions during an NMR experiment, guiding data acquisition, processing and plotting, and even analysis of the data. Key functions are contained in macros and menus that enable the user to acquire and analyze NMR data or simplify complex or tedious operations with a single command.

mail: UNIX includes tools for mailing messages and data between users on a multiuser system or between users of different systems in a computer network. *See also* e-mail.

magneto-optical disk: *See* MOD drive.

maintenance: Little hardware maintenance is required with Sun workstations, but operators should blank or lock the screen when not using it over extended periods of time and clean it when necessary, check the fans periodically (if there are any), and keep the ventilation slots free by removing any dust that might accumulate there.

maintenance contract: In a time of rapidly evolving software, it is important to keep system up to date.Your Sun UNIX and Varian VNMR software is expected to make major progress each year, and a software maintenance contract from Varian keeps both up to date. It should also be kept in mind that UNIX and VNMR really work together. There is no guarantee that the newest VNMR release will work with a much older UNIX release or that the newest UNIX will work with an older VNMR release.

manual: Manuals are important to users in at least two stages of working with the system. The beginner needs the more elementary parts of manuals to get started and to learn about the basics of the software and hardware. Later on, the manuals are valuable to the experienced user, because nobody can learn all the software features by heart. At this stage the online manuals are even more important than those on paper. Both UNIX and VNMR have a `man` command for online help. The UNIX `man` command displays an exact copy of the required descriptions from the Sun UNIX manual if the man pages are loaded. These texts are formatted on the spot using `nroff`. Preformatting of all the manual text, at the expense of several megabytes of disk space, is possible with the `catman` command. This speeds up the display of the manual pages and reclaiming the disk space lost by `catman` can be avoided by deleting the original `nroff` files `/usr/man/man*` afterwards.

mass storage: Device such as a hard disk or tape for storing large amounts of data.

matrix printer: *See* dot matrix printer.

MBus (Memory Bus): Fast bus (over 300 MB per second) used in the SPARCstation 10 and SPARCstation 20. In these machines, the CPU modules plug into the CPU board through a MBus connector. This allows for an easy and cost-efficient upgrade to a system with higher performance, and a second MBus connector allows increasing the number of processors on some SPARCstation models. On high-end Sun Ultra workstations, the MBus expansion port was replaced by a UPA connector.

MC68000: The first 32-bit CISC CPU from Motorola. Some people would call it a 16/32-bit CPU, because it uses 16-bit data I/O, a 24-bit address width, and offers 32-bit register width/32-bit calculations. The MC68000 was used in the Gemini and VXR-4000 host computer, as well as in the UNITY*plus*, UNITY, VXR-S, VXR, and Gemini acquisition computers. The MC68000 was further developed into the 68020, 68030, and 68040 models. *See* MC68020, MC68030, MC68040, Motorola.

MC68020: 32-bit CISC CPU from Motorola, developed from the MC68000. The 68020 offers 32-bit addressing, calculations, and data I/O, and includes a 256-byte primary instruction cache. It operates at 12 to 33 MHz. Integer performance is up to 10 MIPS. For floating point calculations, the MC68881 floating point coprocessor is typically added. The MC68020 powered the Sun-3 workstation family. It was further developed into the 68030 and 68040 models. *See* MC68000, MC68030, MC68040, MC68881, Motorola.

MC68030: 32-bit CISC CPU from Motorola, developed from the MC68020 by adding a 256-byte primary data cache, an improved architecture, an on-chip MMU, and clock rates up to 50 MHz. Integer performance is up to 18 MIPS. For floating point calculations, the MC68882 floating point coprocessor was typically added. The MC68030 powered the last models of the Sun-3 workstation family (Sun-3/80). It was further developed into the 68040 model. *See* MC68000, MC68020, MC68040, MC68881, Motorola.

MC68040: 32-bit CISC CPU from Motorola, developed from the MC68030 by adding an on-chip floating point unit and increasing the instruction and data caches to 4 KB each. The MC68040 is available at clock rates up to 40 MHz. Integer performance is up to 44 MIPS. The MC68040 is used in the UNITY*INOVA* acquisition computer. *See* MC68000, MC68020, MC68030, Motorola.

MC68881, MC68882: Floating point coprocessors for the MC68020 and MC68030 CPUs from Motorola. *See* coprocessor, MC68020, MC68030, Motorola.

megabyte (MB): 1,048,576 bytes, which corresponds to 1024 kilobytes (KB).

megaflops (MFlops, million floating point operations per second): An older performance measure for computers. The MFlops rating is determined by either the Linpack (typical for Fourier transformations) or the Whetstone (typical for most other tasks) benchmark test. Whetstone values are typically 6 to 8 times larger than Linpack. Because most operations in VNMR occur as floating point calculations, the performance of a computer in MFlops is more important for VNMR users than the computer speed measured in MIPS; however, the MFlops specifications are gradually being replaced by the SPECfp suite of performance tests. *See* SPEC, SPECfp.

menu: Software assistance by offering a list of options. Usually the selection of an option is made by pressing a function key, entering a character or sentence, or by selecting an option with the mouse. VNMR uses an elaborate menu system that allows for a wide variety of actions through menus. Menu selections in VNMR can be interspersed with keyboard commands. Despite its power, the structure of the menu system is simple and safe. It offers optimum help for both the beginner and the experienced user. In addition, the menu system is flexible and can be modified by any user. *See* menu button.

menu button: With mouse-oriented user interfaces, like on Sun workstations, menu options often are button-like fields on the screen that are selected either by function key or by moving the mouse on the selected field and then pressing the left mouse button.

mezzanine board: A piggyback-type expansion board that plugs into a motherboard or expansion board via standard connectors. Mezzanine boards are mounted flat on a motherboard and are often used for VME board expansions. The UNITY*INOVA* pulse programmer can be expanded via a mezzanine board. *See* VME, piggyback board.

MFlops: *See* megaflops.

microprocessor: Complex CPU chip that powers personal computers, workstations, server systems, and even many large scale computers. A microprocessor is almost a little computer of its own in that it combines many components that before were placed in separate chips. Only the advances in chip technology and the growing degree of integration make it possible to put all this into a single integrated circuit. Twenty years ago, the first popular microprocessors were 8-bit processors such as the 8080 and the Z80. The current top-of-the-line models are 64-bit processors. Early models essentially consisted of an integer calculator (the arithmetic logical unit, ALU), a small set of registers, and I/O control functions. Modern microprocessors integrate several million transistor functions on a single chip and include several integer units for parallel execution, a large set of registers, one or several floating point units (FPU), a set of primary cache resisters, a memory management unit (MMU), and a controller for external cache. They not only operate at clock rates up to 200 MHz and more, but they also execute several commands in parallel. *See also* CPU, ALU, FPU, MMU, 16-bit computer, 323-bit computer, 64-bit computer.

MicroSPARC: Compact version of the SPARC CPU chip used in the Sun SPARCstation 4 and the SPARCstation 5. The MicroSPARC chip includes floating point processing, 6-KB primary cache, and memory management (MMU) functions on one chip. The current implementation runs at 110 MHz and performs at 1.99 SPECfp95 and 1.59 SPECint95 (65.3 SPECfp92 and 78.6 SPECint92). While offering a system performance comparable to a SPARCstation 10, this chip allows simplifying the CPU board, thus reducing the cost of the workstation. *See also* SPARC, TurboSPARC.

minicomputer: Mid-sized, local computers that were often used in the scientific environment, such as DEC's PDP series and MicroVAX. Most of these systems now been replaced by workstations or even PCs, which generally have become much more powerful than typical minicomputers.

MII (Media Independent Interface): Ethernet connector and cabling standard that permits connecting transceivers for coaxial cabling via AUI connectors, fiber-optic networks, unshielded twisted pair (UTP), using the appropriate connector and transition cable. Most modern Ethernet interfaces offer both MII and RJ45 (for 10/100BaseT) connectors. *See* AUI, Ethernet, UTP.

mini UNIX: Minimal UNIX configuration for installing the full Sun software, loaded into the swap partition under Solaris 1.x. For Solaris 2.x, software is installed from an entirely memory-based version of UNIX.

MIPS (million instructions per second): Mostly determined as VAX-MIPS, where the performance of a VAX 11/780 (running a specific program, the so-called Dhrystone integer performance test) is 1 MIPS. The Dhrystone and MIPS specifications has been replaced by the SPECint performance test. *See* SPEC, SPECint.

MIPS: Manufacturer, recently acquired by SGI, of RISC CPU chips. Early MIPS models included the R2000 and R3000 32-bit CPUs used in SGI graphics and other workstations. The current models are all 64-bit RISC CPUs: the R4000, R4400, R4600, R5000, R8000, and R10000. These power the SGI Indy and Indigo2 workstations as well as some high-end SGI computers. *See* 32-bit computer, 64-bit computer, CPU, SGI.

MMU (memory management unit): Hardware function used to obtain virtual memory capability, which allows handling of very large data tables with limited RAM. On current SPARC CPU chips, this function is integrated into the CPU.

MOD drive: Magneto-optical disks are removable optical disk media (similar to the CD-ROM) that can be rewritten many times. The term “magneto-optical” describes the working principle of these storage media: for writing such disks a strong laser beam locally heats the surface in a magnetic field, locally changing the magnetization of the information layer. The magnetization changes the local optical properties of that layer, such that a weak laser beam can read back the information. Even though MOD disks are very versatile storage media with a capacity of up to 1 GByte or more per side, no format standard for MOD disks was ever defined (which lead to a diversity of incompatible formats), and no major manufacturer built such drives. The DVD-RAM may soon replace the MOD drives. *See* CD-ROM, DVD, DVD-RAM.

modem (modulator-demodulator): Transmission of binary data, most often via telephone lines, by modulation of a low frequency, usually below 10 KHz.

monitor: As hardware, a monitor is a CRT screen for displaying text and graphics information. As software, the monitor level is a built-in PROM-based diagnostics and start-up software on Sun workstations. When a Sun is switched on, the monitor diagnostic is called and started automatically. It resets the hardware, does a series of hardware tests, such as testing memory, and calls a boot program to boot up UNIX. Before switching off a Sun workstation, the system should be run down to the monitor level using the `init 0` command, which makes sure all software is terminated properly.

Mosaic: The first graphical Web browser. While hypertext and HTML had already existed for some time, the early browsers were ASCII-based only. Mosaic was developed at the NCSA (the National Center for Supercomputing Applications) and became commercially available in 1993, drawings millions of user to the Web. More recently, Netscape, a more advanced graphical browser that has originally grown out of the same NCSA environment, has become more popular than Mosaic. *See* WWW, Web browser, Hypertext, HTML, Netscape.

motherboard: The main CPU board, especially on computers that in essence consist of a single printed circuit board. The term was probably chosen because a motherboard often houses a number of additional plug-in extensions, often called daughter boards.

Motif: X Window System graphical user interface as proposed by the OSF (Open Systems Foundation). Motif is used by IBM and other companies. The Sun equivalent to Motif is OpenLook. *See* X Window System, OpenLook, CDE.

Motorola: Computer and chip manufacturer that manufactures the MC680x0 family of CISC CPU chips and the PowerPC series of RISC CPU chips, such as the PowerPC 601, the PowerPC 603/603e, the PowerPC 604/604e, and the PowerPC 750 (“G3”). *See* MC68000, MC68020, MC68030, MC68040, MC68881, CISC, RISC,

mount directory: Normally empty directory on which other disk partitions or directories can be mounted. The mount process hides the contents of the mount directory such that after mounting, the `ls` command displays only the files of the file system that have been mounted, not those of the mount directory. *See* mounting.

mounting: UNIX facility that allows making other file systems part of the currently accessible file system. The file systems are normally entire partitions on the same disk, on other disks, or even on a remote or server disk. After mounting, these partitions can be accessed like any other part of the file system on which they are mounted. The `/usr`, `/export/home`, and other disk partitions must be mounted before they can be accessed. The mounting normally happens automatically during the bootup procedure.

mouse: Hand-held unit with one or several buttons that is moved around on the desk or on a special board for positioning a cursor on the screen. Pressing a mouse button is called clicking. Pressing a mouse button and then moving around the mouse while holding the button down is called dragging. The mouse detects movements either mechanically through a ball or rollers or optically through reflection of a light source on the mouse. Optical mice work only when moved on a special pad placed in landscape orientation. Mechanical mice operate on almost any surface. Sun has recently switched from an optical mouse to an mechanical mouse.

mouse buttons: Sun software requires a mouse with three buttons. The left button (when the cable is pointing away from the user) is the main button, which activates functions by clicking on them. The right button is usually the menu button, used to call pop-up menus, but it might also be used for other purposes. The middle button is usually used to drag graphics around or to set a size of some displayed item.

mousepad: A surface for the mouse to move on. Required for optical mice, but optional for mechanical mice. *See* mouse.

MPEG (Motion Picture Experts Group): Standard file format for compressing and storing movies as digital video data. MPEG compression can be used in multimedia Web applications and Web pages, although currently for many Web users the Internet access speed is to slow to allow for good quality online video transmission, even with MPEG. Software that plays MPEG movies is not standard on most systems yet, therefore most Web browsers only download MPEG files for playback later. *See* HTML, JPEG.

MS-DOS (Microsoft Disk Operating System): Standard operating system for IBM PC personal computers and PC clones. MS-DOS-based software can be run on a Sun by running optional MS-DOS emulation software. Alternatively, the UNIX file system can be accessed from an MS-DOS-based computer through Ethernet, which requires loading Sun's PC-NFS software on the PC. *See* Ethernet, NFS.

MT: *See* multithreading.

MTBF (mean time between failures): The length of continuous operation expected for hardware, usually in thousands of hours, giving a measure of the reliability of hardware.

multimedia: Applications that present text with animated visual or audiovisual information, such as a Web page using sound and a video player. *See* MPEG, WWW.

multiprocessor system: Computer that uses several CPU boards or CPU chips in parallel. Standard workstations only allow for pseudo-parallel execution of processes by extensive time-sharing on a single CPU. In its simplest form, a multiprocessor system allows for true parallel operation by distributing processes onto the different CPUs. This mode usually does not require changes in the individual programs. Special compilers or programs that are written with multithreading also allow a single process to access several CPUs in parallel. SPARCstation 10, SPARCstation 20, and Ultra 2 workstations can be expanded to a total of two or four processors. *See* multithreading

multitasking system: Operating system that can handle and run several programs simultaneously. This normally includes time-sharing between the different active tasks, with scheduling according to the individual task priorities.

multithreading: Programming technique using special multithreading utilities that takes advantage of multiple processors if present. Programs written without multithreading are not faster if run on multiprocessor systems, while applications written with multithreading can spread their load onto different processors. Of course, even without multithreading, multiprocessor systems can still distribute the different processes over several processors, which certainly helps in multiuser and server situations.

multiuser system: Operating system that allows for several users to be logged in at the same time on the same CPU. Ideally, such an operating system also handles the communication between the various users through electronic mail and other online communication utilities such as `mail` *and* `talk` on UNIX. A multiuser system also implies that a single user can operate several "pseudo-terminals" in separate windows (or on separate terminals) simultaneously.

Navigator: *See* Netscape.

NCSA Mosaic: *See* Mosaic.

Netra: Sun workstation with special hardware and software configuration, preconfigured as a Web server and gateway to the Internet. Netra systems are based on a SPARCstation 5 or SPARCstation 20, configured with Web server and e-mail software, and is set up for an asynchronous PPP link to an ISP, typically via a leased line. Netra systems usually have no user interface, because they are configured and operated through the network. *See* Sun Microsystems, SPARCstation, PPP, ISP.

network: Group of interconnected computers and peripherals communicating with each other.

Netscape Navigator: Currently, the most popular graphical Web browser. Netscape was created by a group of people who developed the Mosaic browser and who formed a new company to market a better browser. While HTML has made steady progress and has evolved over the years, Netscape has always been ahead of the crowd by new features such as tables, frames, and more font and background options. Netscape Navigator was one of the first Web browsers to support Java applets and JavaScript, and they are among the first ones to support Sun's new Tcl/Tk language utility. Netscape Navigator is written in a way that permits adding new facilities easily via plug-in software, such as the Tcl/Tk plug-in. *See* WWW, Web browser, Hypertext, HTML, Java, JavaScript, Tcl/Tk, Mosaic.

NIS (Network Information Service): Sun's database software for networked workstations that facilitates the system administration, including sharing password files, network address files, and host libraries. Prior to SunOS 4.1, NIS was named "yellow pages" (this name still shows up in some command names) but the name was changed due to a trademark conflict.

NFS (Network File System): Sun's high-level protocol for disk sharing via Ethernet. Sun software allows accessing remote disks the same way as local disks. Remote disk partitions are mounted onto local directories and, thereafter, behave like a part of the local file system. NFS has been adapted by most UNIX systems.

node: Individual computer in a network. Every node has a dedicated, unique address and most have a name such as varian, fred, or jane.

non-interlacing: *See* interlacing.

null device: Receives output and then discards it. Sometimes called a bit bucket. In UNIX, the null device is the file `/dev/null`.

null modem: Serial ports, such as RS-232 and RS-423, can be configured in terminal or modem mode. In the connector for terminal mode, line 2 is the transmitting signal and line 3 is the receiving signal. In the connector for modem mode, these pins (as well as all the handshaking lines) are interchanged, such that a straight-through cable can be used. Often, both ports are configured in the terminal mode. In such a case, lines 2 and 3 have to be crossed for proper communication and the handshaking lines must be rearranged. The device that interchanges signal and handshaking lines between two ports in terminal mode is called a null modem or null modem cable. This device can be a cable (but not with straight-through wires in this case) or a little box that is inserted between one of the ports and the cable. If printers, plotters, or terminals are hooked up to the ports on the Sun CPU board, a null modem is normally required. For exact instructions, consult Sun manuals or a Varian specialist. Often, such as on an RS-232 expansion board, serial ports can be configured such that a null modem is not required. The two most common types of null modems are those using software XON/XOFF flow control and those with hardware handshaking. An XON/XOFF null modem has pin connections 1–>1 (optional), 2–>3, 3–>2, and 7–>7. Null modems for hardware handshaking have pin connections 1–>1 (optional), 2–>3, 3–>2, 4–>5, 5–>4, 7–>7, 6+8–>20, and 20–> 6+8. Because the hardware type is fully compatible with the XON/XOFF flow control null modem, the hardware type of null modem is preferred. In rare cases, other special types of null modems with different wiring may be required. Ask Varian specialists for advice.

null modem cable: Cable or device that interchanges transmit and receive lines, and the corresponding control lines, between two equally configured RS-232 ports, both terminal or both modem. Usually a null modem cable consists of a normal RS-232, or V.24, cable and the null modem, a box that is inserted between the cable and one of the two ports. *See* null modem, RS-232.

NVRAM (non-volatile RAM): Battery-buffered RAM used to store system configuration and bootup parameters. Earlier workstations used EEPROMs instead. *See* EEPROM.

object file: A compiled program module, typically a file with a "`.o`" extension. In software that consists of many source files or modules, each source file is usually compiled separately, and an object file is created for each of them. These object files then are linked with each other through a linker, and with additional modules from C libraries and other libraries, such as graphics functions, in order to create an executable file. *See* OOPS, class.

object-oriented programming style: *See* OOPS.

objects: Essential element within the object-oriented programming style (OOPS). Objects in programming have internal properties, consisting of private elements that cannot be accessed by the calling program, and well-defined external properties, consisting of public elements that can be accessed by the calling program. An object is an instantiation of a class. The class defines the properties of the object, such as a type definition. An object might be something like a complex variable of the type class. Objects exist in the computer memory for a well-defined lifetime, and it is possible to define operations that are performed upon construction and destruction of an object. *See* OOPS, class.

off-line: Submitting a task for processing in which control over the task and timing is transferred to the operating system. Off-line tasks in UNIX include mail, printing, and plotting.

OLIT (OpenLook Intrinsics Toolkit): Application programming interface (API) used for creating applications under OpenLook (the Sun X GUI). *See* API, OpenLook.

online: A process under direct control and access of the operator. Disk-based manuals are online help for the operator, and any processing invoked and controlled at run-time by the operator is called online.

OOPS (Object-Oriented Programming Style): Programming style that uses objects. The objective of OOPS is to make programming safer, easier, and more portable. It makes programs easier to maintain, and it facilitates cooperation between programmers and programming groups because it enforces clear definitions. Object-oriented programming is facilitated, but not necessarily enforced, in certain programming languages such as C++ and Java. *See also* object, class, C++, Java.

open access: Any user can access a file. Under UNIX, for security reasons, open access files are limited to specific disk partitions. *See* permission.

OpenGL: Application programming interface (API) for creating interactive 3D graphics visualization, simulation, and animation applications, such as molecular modeling software. OpenGL was introduced by SGI and was one of the ingredients that made SGI workstations popular in the scientific visualization market. OpenGL is now becoming available on Sun platforms (Solaris OpenGL). The first version is for systems with Creator3D graphics only, but later versions will run on any platform (with lower performance). Solaris OpenGL should bring a new range of software products, such as molecular modeling, onto the Sun platforms. *See* API, SGI.

Open Look: Sun's original X Window system graphical user interface, part of Sun's OpenWindows software package. As OpenWindows is replacing SunWindows, Open Look is replacing SunView, starting with Solaris 2.0. *See* OpenWindows, X Window System.

Open Systems Foundation: *See* OSF.

OpenWindows: Sun's complete X Window System software package, containing the OpenLook graphical user interface. OpenWindows replaces SunWindows, which is no longer available with Solaris 2.0. *See* Open Look, CDE.

operating system: A program, such as UNIX, DEC's VMS, or Microsoft's MS-DOS, Windows, or Windows 95, that manages the resources of a computer by handling process scheduling, input and output procedures, interprocess communication, file management, and other housekeeping duties.

operator interface: Devices that link the computer and the operator including the screen, keyboard, and mouse.

operator overloading: In most programming languages, math and logical operators, such as +, –, *, /, and parentheses, only work on a restricted and well-defined set of operands. For example, math operators require integers or floating point numbers. In C++, it is possible to define what should happen when such operators are applied to unusual operands, such as user-defined variable types, complex variable types such as entire arrays and structures, or objects. *See* C++.

OSF (Open Systems Foundation): Organization founded to counteract the UNIX International (UI) initiative Both OSF and UI are competing in the creation of a standard to succeed System V UNIX. Members of OSF are IBM, DEC, HP, and other companies. OFS's X user interface is Motif. *See* Unix International, Motif.

OSF/Motif: *See* Motif.

overloading: *See* function overloading, operator overloading.

owner: Assigned to every UNIX file is an owner, normally the person that created the file. The name of the owner can be changed with special commands that can only be performed by the superuser.

packet: Large amounts of data are not transferred at once over Ethernet, but only in small, well-defined units called packets. If the transfer of any packet fails, perhaps because of a collision with another packet, Ethernet automatically sends it again.

paging: The loading and off-loading of individual pages of 4 or 8 KB between RAM and the swap space on a virtual memory system.

parallel computing: Standard workstations allow only pseudo-parallel execution of processes by extensive time-sharing on a single CPU. True parallel computing is available on some multiprocessor computer systems or through special software that uses free CPU time in a network, such as the Varian 3D NMR processing software. It seems likely that multiprocessor systems will lead the top-end workstation market in the near future. CPUs that operate at high clock rates (100 MHz and more) can be built into faster workstations, but it turns out the faster clock rates and technologies, such as ECL, require special components and circuitry throughout the CPU board, making a system too expensive. It is cheaper to use two or more CPUs on a single CPU board. This speeds up multitasking systems immediately, but not necessarily single processes, because special compilers and software are required to make a single program use efficiently several processors in parallel. *See also* multiprocessor system.

parallel I/O: Simultaneous transmission of multiple bits over separate parallel lines. All bus structures used in a Sun—VME, SBus, MBus, UPA, SCSI and other—use parallel I/O. Sun workstations also have a Centronics parallel port for printers, *see* Centronics.

parent: The directory above the current working directory. In UNIX systems, the parent level is accessible under the name "`..`" (two periods).

parity checking: Error checking, but not correction, implemented on most RAM memories for detecting 1-bit-per-byte errors. *Compare with* ECC memory.

partition: *See* disk partition, disk slice.

Pascal: High-level programming language that is excellent for teaching purposes because it educates programmers to write clear and logically structured programs easy to read for other people. Pascal can also be used for technical or scientific programming, but lacks flexibility compared to C and C++. There are Pascal compilers, Pascal interpreters, and P-code (semi-compiled Pascal) interpreters. *See* interpreter.

password: Secret keyword that is typed, without being displayed, after entering the user name to the login shell. Although passwords are optional, a secure password keeps data protected, because only the owner and the superuser have access to the data. Changing the password frequently is recommended. Longer passwords (up to 8 characters are significant) that contain numbers and uppercase and lowercase characters are more difficult for others to find out or guess.

PCI (peripheral component interconnect) bus: 32- or 64-bit standard bus structure used in most PCs. The PCI bus operates at 33 MHz or, more recently, at 66 MHz. Because many more PCI expansion boards are sold than SBus boards, PCI boards are less expensive. On the other hand, SBus cards are more compact. Starting with the Ultra 30, Sun is switching from the SBus interface to systems with PCI bus. The PCI bus offers transfer rates of 50 MB per second (32-bit bus width, 33 MHz) up to 200 MB per second (64-bit bus width, 66 MHz). See Ultra 30, SBus.

PCMCIA (Personal Computer Memory Card Interface Association): A standard that defines the layout and the electrical interface of external memory extension cards. Developed for use in portable computers, PCMCIA cards are very small and are used for portable memory cards, modems, serial port interface cards, and even Ethernet ports. PCMCIA slots can be added to Sun workstations via a SBus card.

PDF (Portable Document Format): PostScript-based file format created by Adobe that is widely adopted for online documentation and for transmitting desktop publishing (DTP) documents. Files in PDF format typically have a "`.pdf`" extension and are viewed using Adobe Acrobat Reader software. VNMR online manuals are now delivered in PDF format. *See* Adobe, Acrobat Reader.

performance: Computer performance is widely discussed because computing equipment power serves as a type of status symbol in today's scientific and industrial communities (as much as MHz does for NMR spectroscopists!). More than that, performance figures, besides prices, are the most important sales argument in computer industry. It has to be recognized, however, that it is very difficult to establish a reliable measure for computer performance. In fact, it is difficult if not impossible to predict how fast a program will run on a specific computer, no matter how many performance figures are at hand—the performance depends greatly on the structure of a program, whether it fits into memory, whether the data it uses are structured, whether it requires considerable disk access, whether the disk access is sequential or randomized, and even small details such as how it uses the CPU registers and how much branching its object code contains. With multitasking systems, it is also important how quickly and easily a system can switch between different programs and how it reacts to loaded situations. Predicting the performance is even more complicated on multiprocessor systems, because it also matters whether and how well a program uses parallel processing. *See* performance measures.

performance measures: Because of the large variety of computing requirements, there are many measures for computing performance. Some tests discussed in this manual are MIPS, a test for the speed of integer calculations compared to a VAX 11/780; MFlops, a test for floating point calculations; SPECint and SPECfp, a series of tests for both integer and floating point performance; and transactions per second, a test for multiuser and network server throughput, mainly for database applications. More tests exists to quantify the disk performance from maximum and sustained transfer rates, to network server performance using NFS and other protocols, and to specify the speed of computer graphics hardware from 2D vectors per second, 3D vectors per second, shaded polygons per second, and so forth. *See* MIPS, MFlops, SPEC, performance.

peripheral: Any hardware involved with input or output including monitors, terminals, and printers.

Perl: Interpreted programming language that is optimized for processing text strings and files. Perl is often the language of choice for CGI scripts, even though it is not bundled with most UNIX operating systems. *See* CGI.

permission: UNIX distinguishes owner, group, and all other UNIX users as levels of access to a file. For each level, UNIX has the read, writer, and execute permissions. Each of these nine permissions can be set individually by the owner or by the superuser.

PGX (PCI-based GX graphics): Fast single-chip 8-bit color graphics accelerator fast windowing and 2D/3D wireframe graphics. Derived from the GX and TGX graphics accelerators, the PGX graphics controller connects to the PCI bus and provides screen resolutions up to 1280 × 1024 or 1600 × 1000. It is standard in Ultra 5 and an option for higher-level Ultra workstations. *See* GX, TGX.

piggyback board: Small printed circuit board mounted on top of a larger board. Early piggyback boards were soldered onto the larger board, but piggyback boards (such as graphics controllers, SBus, MBus or UPA expansion cards) now usually plug into a connector on the main board.

pipe: UNIX facility that permits taking the output of a program and, instead of sending it to the display or other output device, feeds it directly into another program. The pipe is called by a vertical bar between the commands, for example, `dmesg | grep Ethernet`. The data do not show up on disk—the two programs are synchronized with each other and use buffers to hold the data. Multiple pipes can be used, and the amount of data transferred via pipe is not limited.

pixel: Single displayed point. On most Sun workstations, a pixel represents 1 bit of display information for black and white, or 8 bits for 256 colors. High-performance graphics accelerators use 24 bits for 16,777,216 colors, or even more bits per pixel, such as for hidden surfaces in 3D graphics.

pizza box: Popular name for the CPU module of desktop SPARCstation, which is about as size of an American pizza box but not as flat.

plotter: Device for putting continuous information, such as drawing lines and text characters, onto paper. VNMR currently supports a series of HP plotters (7475, 7550A, 7550+, DraftPro, and DraftMaster), plotters that emulate an HP plotter (such as the LaserJet III, 4 and 5, and Lexmark printer), and the Nicolet Zeta-8 plotter (in HP7475 emulation mode).

port: Connector for bus extensions or I/O devices.

portability: The ability to take a program, or software in general, from one computer to another computer and run it there. Normally, the software has to be recompiled for that purpose. Compilers in general are not portable.

PostScript: Standardized page description language from Adobe, used mostly with laser printers. PostScript allows the description of complex page layouts with simple commands. Lines, circles, and other vector-based objects can be described with a few PostScript commands instead of sending a dot matrix, often several million bytes, to a printer. PostScript also features output independent from the technical characteristics, including resolution and aspect ratio, of the output device. PostScript is also used for displaying graphics on the screen and sending complex documents over a communications link.

power failure: Interruption in the line voltage for more than about 0.1 second, stopping CPU execution, causing loss of RAM memory contents, and corrupting data files. The files most affected are those files partially in buffer memory that were just about to be written onto disk. After the interruption, the CPU does an automatic start-up, checks the hardware and disk file systems, and boots up UNIX, ending with a login prompt.

power switch: Turn on power switches on a Sun workstation in this order: screen, CPU, and hard disk (if a separate unit). Turn off power switches in this order: shut down the software properly, switch off the hard disk (if a separate unit), the CPU, and the screen. With the exception of the screen, which can be switched off overnight, avoid excessive switching on and off because it causes thermal stress in the computer hardware.

PPP (Point-to-Point Protocol): Protocol that allows building up IP (TCP/IP) connections over serial lines, modems, and ISDN connections. This not only offers the advantage of error checking and correction, as opposed to a link established by a simple modem dial-up connection, it also permits building up a temporary full Internet connection over serial lines and modems. A similar but somewhat less popular protocol is SLIP. *See* serial I/O, SLIP, TCP/IP, Internet.

precision: Length of a data word. Sun workstations are 32-bit or 64-bit computers, numeric information is stored in 16-bit, 32-bit, or 64-bit words: integers are normally stored in 32-bit data words (`int` or `long` in C), giving 32-bit precision, or in 16-bit “short” data words (`short` in C). Floating point numbers are stored in 32-bit (`float` in C) or 64-bit format (`double` in C).

printcap: UNIX file `/etc/printcap` containing special codes that define how all possible printers must be addressed, including baud rate, handshaking, and translations (e.g., into HP/GL, or data compression). Only used in BSD (`lpr`) printing. The file `printcap` still exists in Solaris 2.x, but is no longer really used.

printer: Device for putting discrete information—characters, dots, dot matrix graphics—onto paper. VNMR supports ink jet and laser printers.

process: Single execution of a program, involving machine-interpretable code and associated data. Every process runs in its own address space, separate from other processes. Several processes run simultaneously in a time-sharing system such as UNIX.

PROM (programmable ROM): *See also* ROM.

prompt: Sequence of one or more characters that indicate where text can be entered from the keyboard. Prompts can include information such as the host CPU name, user name, command line number, and more. Sun UNIX and VNMR software have a few conventions for the prompt: # is the prompt for single-user mode, *`hostname`*`#` is the prompt for `root`, *`hostname`*`,`*`username`*`,`*`command_line_number`*`>` is the prompt for a normal NMR user, and no prompt is present in the VNMR command entry window. The default shell prompt is *`hostname`*`%` for the C shell (when starting a C shell with `csh -f`, without running `.cshrc`), and `$` for the Bourne shell.

protocol: Conventions for communication over interfaces and buses. Protocols for Ethernet include TCP/IP, NFS, and limNET. On Ethernet, several protocols can coexist on the same network, but only machines that use the same protocol can talk to each other.

QIC (quarter-inch cassette): The standard definition for 1/4-inch cassettes and tape drives. Individual standards include QIC-11 (4 tracks, 26 MB with 600-foot tapes), QIC-24 (9 tracks, 60 MB with 600-foot tapes), QIC-150 (24 tracks, 150 MB with 600-foot tapes), and QIC-2000 (2.5 GB). Sun-3 and early Sun-4 systems were equipped with QIC-24 tape drives that could also read and write QIC-11 tapes. Early SPARCstations are equipped with QIC-150 tape drives. They can also read but not write QIC-11 and QIC-24 tapes. Currently, QIC-2000 drives for Sun workstations compete with DAT (5 GB) and Exabyte (14 GB) tape drives, *see* DAT, Exabyte.

quit: *See* exit.

RAID (redundant arrays of inexpensive disks): RAID stands for a number of concepts (RAID-0 through RAID-5) for improving the performance and security aspects of disk subsystems. Both these aspects have gained importance over the recent years, especially in industrial environments, and some of the RAID concepts (RAID-0, RAID-1, RAID-5) have become available for Sun users through the Solstice DiskSuite software and disk arrays hardware. *see* RAID-0, RAID-1, RAID-3, RAID-4, RAID-5

RAID-0: Concept of disk striping: For sequential disk I/O (*not* for random disk I/O), the performance of the individual disk drive is the main bottleneck. Therefore, much higher throughput can be achieved if the data are spread over multiple disks that are then accessed *in parallel*. If this is combined with a fast/wide SCSI interface, transfer rates of up to 20 MB per second can be obtained. *See* RAID, RAID-1, RAID-3, RAID-4, RAID-5.

RAID-1: Concept of disk mirroring, While RAID-0 deals with the disk performance, RAID-1 deals with security. In fact, RAID-0 increases the system vulnerability, because the failure of any single drive would affect the integrity of the data on all the disks. With RAID-1, every disk is mirrored with a second disk, and data are always written twice. While this concept is rather expensive because it doubles the number of disks, it has the advantage that any single disk failure has no effect on the data integrity. Some hardware with RAID-1 support even allows unplugging and replacing disk drives while the system is running (so-called hot spares). Because RAID-1 increases disk I/O and lowers the performance, it is often combined with RAID-1. This is then called RAID-0/1 or RAID-0+1. *See* RAID, RAID-0, RAID-3, RAID-4, RAID-5.

RAID-3: Uses disk striping (RAID-0) and tries to increase its security by adding a disk containing parity information, typically one parity disk per 4 data disks. Therefore, RAID-3 adds security without the costs of full disk mirroring, as used in RAID-1. A problem with this setup is that the parity disk becomes the system bottleneck, because every data write operation also requires a write operation on the parity disk. Also, RAID-3 is implemented at a byte level. This has the disadvantage that with virtually every I/O every disk in the array is involved. This can be overcome with RAID-5. *See* RAID, RAID-0, RAID-1, RAID-4, RAID-5.

RAID-4: Almost identical to RAID-3, except that the implementation is done in blocks instead of bytes. This allows satisfying most I/O requests from a single disk, but it still suffers from the single parity disk as a bottleneck for write operations. *See* RAID, RAID-0, RAID-1, RAID-3, RAID-5.

RAID-5: Similar to RAID-3 and RAID-4, except that the parity information is spread over all disks of the array. This balances the load, combining the security of RAID-1 and the performance of RAID-0 without the costs of RAID-1. *See* RAID, RAID-0, RAID-1, RAID-3, RAID-4.

RAM (random-access memory): Memory that the CPU can read and write to. RAM is unlike ROM, which is also random access but read-only memory.

recursion: Calling the same program or a subroutine from within itself. Under UNIX, recursion also refers to programs that recursively cover a directory and all of the subdirectories and subfiles below the directory. For example, the `rm` command with the `-r` recursive options, `rm –r /stuff,` removes the directory named `/stuff` and all subdirectories and subfiles below `/stuff`. *Be very careful when using the recursive option with UNIX commands.* `rm -r`, for example, can delete many files, including files you don’t want to delete, in a fraction of a second.

refresh rate: Rate at which the display is refreshed. The higher the refresh rate, the less flickering is obtained. Sun screens use a 66- or 76-Hz refresh rate. *See* interlacing.

release: To be useful, software must constantly evolve, but not all evolution is straightforward. The implementation of new features and sometimes even repairing old bugs can create new problems. Adding in the need to keep software costs under control makes manufacturers collect software in packages and only release such packages when they are relatively consistent, reasonably tested, and an advancement over the previous software. New software packages are called releases or revisions, and they are normally labeled with a release or revision number and a date. Since most bugs are inherently linked to the software release in which they are found, it is absolutely essential to report the software revision number when bugs are reported.

remote: Actions performed on or invoked by other computers.

revision: *See* release.

RISC (reduced instruction set computer): Reduces the number of instructions at the machine level of a CPU to make the processor simpler and usually faster. Most graphics workstations, including the SPARCstation and Ultra series computers, use a RISC processor. *Compare with* CISC.

RJ45: Easy to plug and unplug connector type used in telephone networks and in 10baseT (10 Mbps) and 10/100baseT (10 and 100 Mbps) Ethernet connections over shielded twisted pair cabling. *See* 10baseT, 10/100baseT, Ethernet.

ROM (read-only memory): Hardware memory that generally cannot be altered by the computer and does not lose its contents when the power is switched off. Most computers use ROM memory to permanently maintain some low-level software such as boot-up routines and system diagnostics. Types of ROMs include PROMs, which are programmable ROMs; EPROMs, which are erasable programmable ROMs; and EEPROMs, which are electrically erasable programmable ROMs. *See* PROM, EPROM, EEPROM.

root: Name of the UNIX superuser, the system manager who has access to all files and processes on the system and does not require any permissions on the system. Also, root is the name for the origin of the file system that is accessible by the slash character (`/`). The root partition is the disk partition `rsd0a` that contains the origin of the UNIX file system as well as the most important UNIX administration data and files, such as the kernel.

root menu: Menu on most graphic screens that pops up when the right mouse button is pressed with the mouse pointing to the screen background. This menu allows calling commands without typing in a shell. Some commands available from the root menu include standard and scrollable windows, special tools such as a screen blanking program, an icon editor, a mail tool, and a program that terminates the windows environment. The root menu can be customized and may contain submenus.

root partition: *See* root.

RS-232, RS-423: Standard interfaces for serial communication. RS-232 technically is a subset of RS-423. RS-232 is specified up to 19,200 baud, RS-423 allows for higher transfer rates (49,500 baud and more). The RS-232 standard defines terminal and modem ports for transmitting and receiving devices, respectively. The assignment of the most important pins in a terminal connector is as follows (“in” means that on this line a terminal port is listening): pin 1 is frame ground (cable shielding); pin 2 is TxD, transmit data (out); pin 3 is RxD, receive data (in); pin 4 is RTS, request to send: (out); pin 5 is CTS, clear to send (in); pin 6 is DSR, data set ready (in); pin 7 is GND, ground; pin 8 is DCD, data carrier detect (in); and pin 20 is DTR, data terminal ready (out). *See* null modem, XON/XOFF, handshaking.

safety: In connection with complex multiuser operating systems, safety includes the protection of files and programs against unintended loss, against data corruption by hardware or software problems, and against access by unauthorized users. UNIX has various ways of protecting files and data.

SBus: Internal bus in desktop SPARCstation and Ultra workstations that uses 32-bit parallel, synchronous data transmission at a speed of over 50 MB per second. The SBus can be accessed though connectors on the CPU board, but one or two of these connectors are used by the graphics controller on some systems. The 64-bit version of the SBus allows for peak transmission rates of 160 to 200 MB per second.

scheduler: Program or operating system utility that assigns CPU time segments from fractions of a second up to a few seconds to the various programs running simultaneously in time-sharing operation in a multitasking system. The scheduler respects program priorities but also tries to avoid CPU blocking by excessively long processes. Except for programs with very high priority, programs are interrupted every now and then, typically once in a second, to enable small processes to get their share of CPU time.

screen: CRT display.

screen burn-in: The luminosity of a CRT display decreases in places where a bright picture is left on for a long time. Activate the screen lock from the root menu or the screen blanking mode to make the screen black, or simply switch the monitor off overnight, over the weekend, and during extended idle periods.

screen blanking: Sun software can automatically turn off the screen after a period of disuse. The screen then turns back on when the mouse is moved or when the keyboard is touched. No password protection is provided. Screen blanking must be user-activated by enter the `xset s on` command once after starting the X environment. This command can be automated by adding it into a start-up script such as `.xinitrc`.

screen locking: A Sun system can be placed in a locked mode, in which the screen is blank and the mouse and keyboard cannot be used, by selecting Lock Screen from Utilities in the background menu. To unlock the system requires a password.

scroll bar: If the displayed information in a window exceeds the size of the window, a program such as `more` is required to scan though the information page by page or line by line or a special window is required that can keep all the information in a buffer. Such windows allow scrolling forward and backward through the information (`more` only allows forward scrolling, unless called on a file). The tool that allows scrolling with the mouse is called the scroll bar, a bar along one edge of the window that symbolizes the size of the displayed information and an indicator where within the entire buffer the current displayed information is located. On OpenLook, scrolling is achieved by moving the mouse onto the scroll bar and dragging the elevator symbol in the scroll bar up and down.

SCSI (Small Computer Systems Interface): Standard interface for 3.5-inch and 5.25-inch hard disks. On Sun-3 and early Sun-4 systems, the SCSI bus performed at less than 1 MB per second; the SPARCstation 2 reaches 1.5 MB per second (actual throughput), and the SPARCstation 10 allows for transfer rates of up to 10 MB per second. Sun Ultra workstations with Creator graphics accelerator are equipped with a fast, wide SCSI interface that allows for transfer rates of up to 20 MB per second.

sector: Part of a circular track on a single disk surface. The number of sectors per track is in the order of 15 to 40. *See* cylinder, interleave factor.

seek time: Time required to access a track on a disk, a function of the distance between the last and the current track. Current hard disks have average seek times of 8 to 15 ms. By working in cylinder groups, Solaris can reduce seek times when reading single files, because such files are not scattered over the entire disk.

serial I/O: Data input and output, bit by bit, via a single line. Most serial interfaces use one line for sending and one for receiving. More lines may be used for grounding, handshaking, and power. Most printer interfaces use 3 to 5 lines, some terminals require 8 lines, and even more are sometimes required for synchronous transmission, *see* RS-232.

SGI (Silicon Graphics, Inc.): Manufacturer of graphics workstations such as the IRIS Indigo, Indy, and Indigo2, all based on the MIPS family of microprocessors. SGI has recently acquired MIPS, a manufacturer of RISC microprocessors. VNMR is available for SGI workstations, *see* VnmrSGI.

server: Computer that provides facilities, mostly disk space, for other computer systems called clients. For the X Window System (X11), the terms client and server have a different meaning. *See* client, X Window System, X client, X server.

SGML (Standard Generalized Markup Language): In the late 1960s, IBM created a Generalized Markup Language (GML) to overcome difficulties in transporting documents between different computers and operating systems. In 1986, GML became ISO 8879 and renamed the Standard Generalized Markup Language. Generalized means that it does not specify how exactly a document is to be presented or formatted, but it instead defines the document type and presentation styles. An SGML document consists of three parts: the character set and which characters are used to differentiate markup tags from standard text; the document type and which markup tags are accepted in the following document, and the document instance that contains the actual text with markup tags. The three parts don’t need to be in the same document or file. HTML is an *instance* of SGML. A HTML document corresponds to the third part of an SGML document, and its markup tags are defined as the hypertext markup language. *See* HTML.

shell: Generally means a platform on which other programs are run under UNIX. Called a shell because when the UNIX functionality is usually visualized, the shells are grouped around the kernel, which is shown as a ball with a small machine-specific core. Shell also stands for a UNIX command interpreter, such as the C shell, Korn shell, or Bourne shell.

shell script: Text file with commands and control statements that are interpreted by a shell. Many UNIX commands are actually powerful shell scripts. Solaris offers different shells, including the C shell, Korn shell and Bourne shell; therefore, it allows for different kinds of shell scripts, using these shell languages. *See* shell.

shielded cable: Cable with grounded electrical shielding against rf pickup from external sources. Such shielding is woven from metal wires or made of metal foil. The shielding can surround many conductors or each conductor can be shielded individually. To avoid ground loops, the shielding should only be grounded on one side of the cable. *See* coaxial cable, ground loop.

shielding: Current computers operate in the range of 50 to 250 MHz (up to 100 MHz in CRT screens), which is in the range of common rf. Therefore, computers can generate rf emission that can potentially interfere with radio and television equipment, as well as analytical equipment including NMR machines that operate within this frequency range. For this reason, there are restrictions on the amount of rf emission from computers. Sun workstations are in fact heavily shielded (the shielding metal case at the same time ensures mechanical stability). At the same time, computers should be protected against rf pickup from external sources. Cables especially can act as antennas and should be shielded. *See* shielded cable.

Silicon Graphics, Inc.: *See* SGI.

SIMM (single in-line memory module): Small boards holding RAM chips that can be mounted easily onto the CPU board without soldering. A SIMM currently can contain up to 64 MB of dynamic RAM, usually with parity checking. High-end workstations, such as the SPARCstation 10 and 20, and Ultras, use SIMMs with ECC. *See* RAM, ECC memory.

single-user mode: Low-level mode of UNIX operation that does not allow for multitasking or daemons. Note that this means no printing is possible. This mode is mostly used for system administration tasks such as checking and repairing file systems, and making tape dumps from disks. *See* multitasking, daemon, multiuser system.

SLIP (Serial Line Internet Protocol): Protocol that allows building up IP (TCP/IP) connections over serial lines and modems. This not only offers the advantage of error checking and correction, as opposed to a link established by a simple modem dial-up connection, it also permits building up a temporary Internet connection over serial lines and modems. A similar protocol is PPP. *See* serial I/O, PPP, TCP/IP, Internet.

SLC: *See* SPARCstation SLC.

slice: *See* disk slice.

SMCC (Sun Microsystems Computer Corporation): A subsidiary of Sun Microsystems, Inc., founded in 1991, that specializes in the development and marketing of new computer hardware. SunSoft now carries out the development and the marketing of new software and of Solaris, the Sun operating system. *See* Sun Microsystems, SunSoft.

SMD (surface mounted device): On conventional printed-circuit boards, chips are mounted by soldering each leg into a dedicated hole in the board with a distance of about 2.5 mm between connections. With the SMD technique, the chip legs are soldered flat onto the surface of a printed circuit board with about two connections per mm. This allows keeping the dimensions of VLSI chips, with their large number of connections, as well as the board itself, small while maintaining high reliability. SMD chips can only be mounted by machines, and it is virtually impossible to unsolder chips. CPU boards in Sun workstations use this technique extensively.

software: Any kind of binary information used in a computer, whether a compiled, program. source code, data, or text files. Software might reside on tape, on disks, in RAM or ROM memory, or on any other temporary or permanent storage device.

software maintenance: Includes file system maintenance and software upgrading. Maintenance includes avoiding overloaded file systems by off-loading data to tape or other file systems or computers, deleting unnecessary files, and checking file systems for consistency. A software maintenance contract guarantees that the software is kept up to date by always providing the latest release.

SoftWindows 95: Software for Sun (Solaris 2.x) workstations that emulates the Microsoft Windows 95 PC operating system and permits running PC software on a Sun workstation. SoftWindows 95 is probably going to replace the WABI software that emulates Windows 3.1 only. As this is pure software emulation, this requires a high performance CPU, such as the newer, PCI-based Ultra workstations. *See* WABI, SunPC.

Solaris 1.1: Identical with SunOS 4.1.3. Solaris is a trademark for operating system software from SunSoft, a subsidiary of Sun Microsystems. Solaris 1.1 is based on 4.2BSD UNIX, but System V UNIX extensions are included as option. Solaris 1.1 is exclusively available on CD-ROM.

Solaris 2.x: Identical with SunOS 5.x, for SPARCstations only. This is the first Sun software release based on AT&T's System V.4 UNIX release, but it also incorporates most 4.2BSD features. Apart from SunOS 5.x, Solaris 2.x also features OpenWindows with OpenLook and CDE (SunView is *not* part of Solaris 2.x). Different from all previous software releases for Sun computers, the C compiler is optional and has to be purchased separately. Solaris 2.x has also been ported to the PC-compatible platform. Solaris 2.x is exclusively available on CD-ROM. The Solaris 2.x versions used with VNMR software include 2.3 (VNMR 5.1 only), 2.4, 2.5, 2.5.1, and 2.6. Sun also makes patches available for Solaris 2.x, and there are also minor releases in-between, such as Solaris 2.5 "Hardware: 1/96", Solaris 2.5.1 "Hardware: 4/97", Solaris 2.5.1 "Hardware: 8/97", Solaris 2.5.1 "Hardware: 11/97", Solaris 2.6 "Hardware: 3/98". Solaris 2.5.1 "Hardware: 11/97" and all versions of Solaris 2.6 are y2000 compliant. Sun workstations come with a Solaris 2.x version that is compatible with the hardware; often, workstations are not compatible with older Solaris releases. Unless you are sure that your workstation is compatible with it, do not retrograde a system with an older Solaris version! *See also* year 2000 compliance.

Solstice: Sun's trademark for a large suite of software products for network administration, system management, telecommunication, PC networking and Internet connectivity.

SPARC (scalable processor architecture): Sun's RISC processor, replacing the Motorola 68020 used in the Sun-3 series of workstations. The original SPARC chip has been replaced by enhanced versions, such as the MicroSPARC, HyperSPARC, SuperSPARC, and UltraSPARC. *See* MicroSPARC, SuperSPARC, HyperSPARC, UltraSPARC. Table 19 lists performance specifications for SPARC processors.

Table 19. Performance Specifications for Sun CPUs

CPU Type	*Integer Performance*			*Floating Point Performance*		
	MIPS	*SPECint92*	*SPECint95*	*MFlops*	*SPECfp92*	*SPECfp95*
MC 68020	1.5 - 7			---		
SPARC	7 – 28.5	up to 21.8		up to 4.2	up to 22.8	
MicroSPARC		59 – 78	up to 1.6		þ47 – 65	up to 2.0
TurboSPARC			up to 3.3			up to 2.9
SuperSPARC		þ45 – 126	up to 3.1		þ54 – 121	up to 3.1
HyperSPARC		þ98 – 194			116 – 226	
UltraSPARC		215 – 330	þ5.4 – þ7.7		300 – 500	þ7.9 – 11.4
UltraSPARC II			10.0 – 16.1			14.9 – 23.5

SPARCclassic: Compact desktop workstation with 50-MHz MicroSPARC CPU (26.4 SPECint92, 21 SPECfp92), 15-inch 256-color 1024 × 768 pixel screen, and 1 SBus slot. *See* SPARCstation.

SPARCstation: Second generation of Sun-4 workstations, based on Sun's proprietary family of RISC CPUs called SPARC, including SPARC, SuperSPARC, MicroSPARC, and HyperSPARC CPUs. The first Sun-4 generation, based on the original SPARC CPU design, consisted of the Sun 4/260 and 4/110. Subsequent Sun-4 workstations are marketed by Sun as SPARCstations, while internally the Sun-4 nomenclature was still in use.

SPARCstation 1: Sun 4/60 workstation, Sun's first desktop SPARCstation, with 20-MHz SPARC CPU (12.5 MIPS, 1.4 MFlops). *See* SPARCstation.

SPARCstation 1+: Sun 4/65 workstation, successor to SPARCstation 1, with 25-MHz SPARC CPU (15.8 MIPS, 1.7 MFlops). *See* SPARCstation.

SPARCstation 10: Family of desktop workstations with 1 to 2 MBus CPU modules (1 to 4 SuperSPARC CPUs), 4 SBus slots, 32 to 512 MB of ECC RAM, up to 2 internal hard disks, optional internal floppy and CD-ROM drive, and built-in ISDN port. *See* SPARCstation, SuperSPARC, MBus, SBus, ECC, ISDN.

SPARCstation 10/30: SPARCstation 10 with 36-MHz SuperSPARC CPU (102 MIPS, 20.5 MFlops; 44.2 SPECint92, 52.9 SPECfp92). *See* SPARCstation, SuperSPARC.

SPARCstation 10/40: SPARCstation 10 with 40-MHz SuperSPARC CPU (110 MIPS, 23 MFlops; 51 SPECint92, 60 SPECfp92). *See* SPARCstation, SuperSPARC.

SPARCstation 10/41: SPARCstation 10 with 40-MHz SuperSPARC CPU and 1-MB cache (110 MIPS, 23 MFlops; 53 SPECint92, 68 SPECfp92). *See* SPARCstation, SuperSPARC.

SPARCstation 10/51: SPARCstation 10 with 50-MHz SuperSPARC CPU and 1-MB cache (135 MIPS, 27 MFlops; 65 SPECint92, 83 SPECfp92). *See* SPARCstation, SuperSPARC.

SPARCstation 2: Sun 4/75, successor to SPARCstation 1+, with 40-MHz SPARC CPU (28.5 MIPS, 4.2 MFlops; 21.8 SPECint92, 22.8 SPECfp92). *See* SPARCstation.

SPARCstation 20: Family of desktop workstations with 1 to 2 MBus CPU modules (1 to 4 SuperSPARC or HyperSPARC CPUs), 4 SBus slots, 32 to 512 MB of ECC RAM, up to 2 internal hard disks, optional internal floppy, and CD-ROM drive. *See* SPARCstation, SuperSPARC, HyperSPARC, MBus, SBus, ECC.

SPARCstation 20/50: SPARCstation 20 with 50-MHz SuperSPARC CPU (134 MIPS, 30 MFlops; 77 SPECint92, 80 SPECfp92). *See* SPARCstation, SuperSPARC.

SPARCstation 20/51: SPARCstation 20 with 50-MHz SuperSPARC CPU and 1-MB cache (133 MIPS, 30 MFlops; 82 SPECint92, 89 SPECfp92). *See* SPARCstation, SuperSPARC.

SPARCstation 20/61: SPARCstation 20 with 60-MHz SuperSPARC CPU and 1-MB cache (167 MIPS, 37 MFlops; 98 SPECint92, 107 SPECfp92). *See* SPARCstation, SuperSPARC.

SPARCstation 20/71SX: SPARCstation 20 with 75-MHz SuperSPARC II CPU and 1-MB cache, 24-bit SX graphics accelerator (205 MIPS, 44 MFlops; 126 SPECint92, 121 SPECfp92; 3.1 SPECint95, 3.1 SPECfp95). *See* SPARCstation, SuperSPARC, SX.

SPARCstation 20/151SX: SPARCstation 20 with 150-MHz HyperSPARC CPU and 512-KB cache, 24-bit SX graphics accelerator (169 SPECint92, 208 SPECfp92). *See* SPARCstation, HyperSPARC, SX.

SPARCstation 330: Sun 4/330, deskside workstation with 25-MHz SPARC CPU (16.0 MIPS, 2.5 MFlops). *See* SPARCstation.

SPARCstation 4: Family of desktop workstations, based on the MicroSPARC II CPU. Color graphics on board, 1 SBus slot, 1 internal disk, mounted on the CPU board, optional internal floppy, and CD-ROM drive. *See* SPARCstation, MicroSPARC.

SPARCstation 4/70: SPARCstation 4 with 70-MHz MicroSPARC II CPU (100 MIPS, 16 MFlops; 60 SPECint92, 47 SPECfp92). *See* SPARCstation, MicroSPARC.

SPARCstation 4/85: SPARCstation 4 with 85-MHz MicroSPARC II CPU (110 MIPS, 18 MFlops; 65 SPECint92, 53 SPECfp92). *See* SPARCstation, MicroSPARC.

SPARCstation 4/110: SPARCstation 4 with 110-MHz MicroSPARC II CPU (135 MIPS, 22 MFlops; 79 SPECint92, 65 SPECfp92; 1.6 SPECint95, 2.0 SPECfp95). *See* SPARCstation, MicroSPARC.

SPARCstation 5: Family of desktop workstations, based on the MicroSPARC II CPU. TGX color graphics on board, 3 SBus slots, up to 2 internal disks, optional internal floppy, and CD-ROM drive. *See* SPARCstation, MicroSPARC.

SPARCstation 5/70: SPARCstation 5 with 70-MHz MicroSPARC II CPU (100 MIPS, 16 MFlops; 60 SPECint92, 47 SPECfp92). *See* SPARCstation, MicroSPARC.

SPARCstation 5/85: SPARCstation 5 with 85-MHz MicroSPARC II CPU (110 MIPS, 18 MFlops; 65 SPECint92, 53 SPECfp92). *See* SPARCstation, MicroSPARC.

SPARCstation 5/110: SPARCstation 5 with 110-MHz MicroSPARC II CPU (135 MIPS, 22 MFlops; 79 SPECint92, 65 SPECfp92; 1.6 SPECint95, 2.0 SPECfp95). *See* SPARCstation, MicroSPARC.

SPARCstation 5/170: SPARCstation 5 with 170-MHz TurboSPARC CPU (3.3 SPECint95, 2.9 SPECfp95). *See* SPARCstation, TurboSPARC.

SPARCstation ELC: Sun 4/25, CPU built into the screen housing, successor of SPARCstation SLC, with 33-MHz SPARC CPU (23.0 MIPS, 3.0 MFlops, 18.2 SPECint92, 17.9 SPECfp92) and black-and-white screen only. *See* SPARCstation, SPARC.

SPARCstation IPC: Sun 4/40, color desktop workstation with 25-MHz SPARC CPU (15.8 MIPS, 1.7 MFlops; 13.8 SPECint92, 11.1 SPECfp92) and 1 SBus slot. *See* SPARCstation, SPARC.

SPARCstation IPX: Sun 4/50, desktop workstation, successor of the SPARCstation IPC, with 40-MHz SPARC CPU (28.5 MIPS, 3.8 MFlops, 21.8 SPECint92, 21.5 SPECfp92), 1 SBus slot, and GX graphics. *See* SPARCstation, SPARC, GX.

SPARCstation LX: Compact desktop workstation, successor of the SPARCstation IPX, with 50-MHz MicroSPARC CPU (26.4 SPECint92, 21 SPECfp92) and 1 SBus slot. *See* SPARCstation, SPARC.

SPARCstation SLC: Sun 4/20, desktop workstation, CPU built into the screen housing, with 20-MHz SPARC CPU (12.5 MIPS, 1.4 MFlops) and black and white screen only. *See* SPARCstation, SPARC.

SPEC (System Performance Evaluation Cooperative): An independent organization that created a test suite for standardized CPU performance tests. SPEC performance figures are now widely accepted as more reliable and comprehensive than MIPS and MFlops numbers. The main objective of the SPEC initiative is to provide a series of tests that cannot be tuned by the computer manufacturers. With the increased number of CPU registers and cache memory, the conventional MIPS and Linpack (MFlops) performance tests produced unrealistically high numbers that had little practical meaning for the performance of everyday programs. Even the SPEC performance test suite is periodically revised to avoid tuning of compilers and computer hardware to the tests. The first test suite was introduced in 1989 as an attempt to reduce the computer performance to a single number, the geometric mean of the execution time of four integer and six floating point intensive programs, compared to the performance of a VAX 11/780. This idea received only limited acceptance, because many users wanted a test of more general UNIX performance involving integer operations, as opposed to the number crunching performance on floating point operations. For this reason, the 1992 test suite consisted of two different figures: SPECint92 was made the geometric mean of the execution time 6 integer performance tests, compared to the performance of a VAX 11/780, and SPECfp92 was made the geometric mean of 14 floating-point intensive performance tests. In 1995, the test suite was further enhanced. The basic tests are partly the same (8 integer performance tests, 10 floating point performance tests), but the test conditions are defined more vigorously, and the new figures were defined, SPECint95 and SPECfp95, based on the performance of a SPARCstation 10/41. For multiprocessor systems, a special set of throughput tests (see SPECrate) was created. *See also* MIPS, MFlops, SPECmark, SPECint92, SPECint95, SPECfp92, SPECfp95.

SPECfp92: Performance measure for floating-point-intensive programs, typical for scientific number crunching. SPECfp92 is the geometric mean of execution times on 14 performance tests, relative to the execution times on a VAX 11/780. *See also* SPEC, SPECint92.

SPECfp95: Performance measure for floating-point-intensive programs, typical for scientific number crunching. SPECfp95 is the geometric mean of the execution times on 10 performance tests, relative to the execution times on a SPARCstation 10/41. *See also* SPEC, SPECint95.

SPECint92: Performance measure for programs dominated by integer calculations, such as typical UNIX commands. SPECint92 is the geometric mean of the execution times on 6 performance tests, relative to the execution times on a VAX 11/780. *See also* SPEC, SPECfp92.

SPECint95: Performance measure for programs dominated by integer calculations, such as typical UNIX commands. SPECint95 is the geometric mean of the execution times on 8 performance tests, relative to the execution times on a SPARCstation 10/41. *See also* SPEC, SPECfp95.

SPECmark: First performance measure created by the SPEC initiative in 1989. The measure consists of 10 tests, including 4 integer and 6 floating-point-intensive programs. The SPECmark performance number is the geometric mean of the execution times for all these tests, relative to the performance of a VAX 11/780. Although the SPECmark may be a more objective measure for general computing performance, it doesn’t reflect the fact that, for example, some programs consist of only integer operations while others are dominated by floating-point calculations. Computer performance can’t be reduced to a single number. It depends heavily on the type of operations involved, whether integer or floating point, or single or double precision; on the structure of the data, whether scalar or vectorized; the amount of disk operations required; and the number of competing processes. To gain more information on integer as compared to floating-point performance, the more recent performance tests (SPECint92, SPECfp92, SPECint95, SPECfp95) have been separated into integer performance, typical for UNIX commands, and floating point performance, typical for scientific number crunching.

SPECrate: Test suite defined by SPEC consortium, designed to measure performance of multiprocessor and single-processor systems. Similar to SPECmarks, SPECrate has separate integer and floating point test suites. *See* SPECmark, MIPS, MFlops.

SRAM (static RAM): Fast but expensive RAM chips, with access times of 60 nsec and less. Due to their higher complexity per storage element, these chips can currently only be made in smaller sizes like 256 kilobits or 1 megabit per chip. On high-end workstations, SRAM chips are used in the cache memory. SRAM does not require refreshing, in contrast to DRAM (dynamic RAM), which can only hold information when it is periodically refreshed. *See* cache memory, DRAM.

stack register: A last-in first-out (LIFO) buffer. When a stack gets full, the items entered first are deleted.

stand-alone workstation: Complete, functional workstation that does not require connections to other computers such as servers or clients.

static discharge: Usually generated by friction when walking around or moving on a chair. Dangerous to any electronic equipment, but can be avoided by humidifying the air and using conducting floors, carpets, and shoes.

STP (shielded twisted pair): Cabling similar to telephone cables, used for 10baseT (standard) and 100baseT (fast) Ethernet, and for ATM for transfer rates up to 155 Mbps. In NMR environments, STP cabling is preferred over UTP (unshielded twisted pair) cabling, because STP avoids rf emissions. *See also* Ethernet, ATM, UTP.

structured programming: A programming style designed to create programs easy to read and understand by persons other than the programmer. Ideally, structured programming is combined with a top-down approach. Overall, both improve programming efficiency. Certain programming languages such as Pascal or C almost automatically lead to structured programs. Low-level programming, such as assembly language, are virtually impossible to structure. *See* top-down programming.

Sun-2: First generation of Sun workstations, which were widely marketed between 1983 and 1986. Sun-2 systems were based on the Motorola MC68010 microprocessor with Multibus boards in the bigger workstations. Performance was below 1 MIPS, but the Sun-2 was already using current graphics standards of 900 × 1152 pixels, 66-Hz refresh rate, and noninterlaced display. Sun-2 was replaced by the Sun-3 series and is no longer supported by Varian as a host computer for their NMR spectrometers.

Sun-3: Sun graphics workstations that succeeded the Sun-2 and were marketed between 1985 and 1989. The Sun-3 workstation series were based on the MC68020 and MC68030 microprocessors from Motorola, and the MC68881 and the MC68882 floating point coprocessors. Larger Sun-3 systems used VME bus boards, with performance values ranging between 1.5 and 7 MIPS. Larger systems could also be equipped with the Sun-FPA, a floating point accelerator board, which resulted in improved floating point performance. Sun-3 workstations are no longer supported by VNMR software. Table 20 compares the Sun-3 models that were formerly used as hosts computers for Varian spectrometers (3/xx systems are not equipped with expansion slots)

Table 20. Sun-3 Computer Models

Model	*MIPS*	*KFlops*	*RAM (MB)*	*Type*
3/50	1.5	88	4	desktop, no color
3/60	3	130	8 - 24	desktop
3/80	3	160	8 - 24	desktop
3/110	2	101	8 - 24	desktop, 3 slots
3/110 + FPA	2	405	8 - 24	desktop, 3 slots
3/150	2	101	8 - 24	deskside, 6 slots
3/150 + FPA	2	405	8 - 24	deskside, 6 slots
3/160	2	101	8 - 24	deskside, 12 slots
3/160 + FPA	2	405	8 - 24	deskside, 12 slots
3/260	4	114	8 - 24^+	deskside, 12 slots
3/260 + FPA	4	465	8 - 24^+	deskside, 12 slots
3/470 + FPA	7	600	8 - 128	server system

Sun-4: Graphics workstations that succeeded the Sun-3 and have been on the market since 1987. The Sun-4, based on the SPARC processor, Sun's proprietary RISC CPU chip, brought a dramatic increase in performance. The first Sun-4 was the Sun 4/260, a deskside workstation with a 12-slot card cage, 8 MB to 128 MB RAM, and performance rated at 10 MIPS and 1.1 MFlops. The next model, the Sun 4/110, was a desktop workstation in a 3 slot card cage (1 slot free) with 8 MB to 32 MB RAM and performance rated at 7 MIPS and 0.8 MFlops. This model was considered the first supercomputing desktop workstation. In 1989, Sun brought out a new generation of Sun-4 computers: the SPARCstations. Modern manufacturing techniques, increased integration levels (the CPU board of a desktop SPARCstation consists of less than 50 chips, excluding the RAM), and modern chip technology allowed Sun to improve the price/performance ratio dramatically. Since 1992, performance has been further boosted through the introduction of newer generations of SPARC CPU chips—the MicroSPARC, HyperSPARC, and SuperSPARC. *See* SPARC, SuperSPARC, MicroSPARC, HyperSPARC, SPARCstation, RISC.

Sun Microsystems: Founded in 1982, Sun is the leader in the workstation market. The keys to its success were SunOS and its integrated networking software, the NFS standard that Sun created, and the SPARC CPU chip technology. In 1991, Sun Microsystems Inc. was restructured by forming a number of subsidiaries such as SunSoft and Sun Microsystems Computer Corporation (SMCC). *See* SPARC, NFS, SMCC, SunSoft.

SunOS 4.x: Sun's UNIX operating system, a converged UNIX. SunOS 4.1.x was based on 4.2BSD UNIX, including System V enhancements. The last SunOS releases are SunOS 4.1.1 for the Sun-3, 4.1.2 for the Sun-4 in general, and 4.1.4 (identical with Solaris 1.2) for most SPARCstations. All currently sold Sun computers are delivered with Solaris 2.x, which contains SunOS 5.x. *See* Solaris 1.1, Solaris 2.x.

SunPC: Expansion board with Intel (486 or higher) processor for Sun workstations (and the necessary software). SunPC permits running Microsoft Windows 3.1 or Windows 95 on Sun workstations, in a Solaris 2.x environment. Because no emulation is involved, the SunPC option performs much better than WABI or SoftWindows 95. There are two different types of SunPC boards: one for SBus-based systems (all SPARCstations, Ultra 1 and Ultra 2 workstations) and a new one (released in summer 1998) for PCI-based Ultra workstations. Microsoft Windows (3.1 or 95) software is not included and must be purchased and installed separately. *See* SoftWindows 95, WABI.

SunSoft: Subsidiary of Sun Microsystems, Inc., founded in 1991, specializing in the development and the marketing of Solaris operating system, which they have ported to the PC-compatible platform. SunSoft also develops Sun's compiler technology and creates new software for the Solaris environment. *See* Sun Microsystems, Solaris 1.0, Solaris 2.

Suntools: Earlier name for SunView. *See* SunView.

SunView: Sun's first proprietary graphical user interface, part of the SunWindows software package. An earlier name for SunView was Suntools. SunView has now been replaced by OpenLook, Sun's X Window System graphical user interface. SunView has not been ported to Solaris 2.x. *See* graphical user interface, SunWindows.

SunWindows: Sun's first proprietary window software. Although very powerful in performance, it can only address local displays and has not become a standard in computer industry. SunWindows has been replaced by OpenWindows, Sun's X Window System software package. SunWindows has not been ported to Solaris 2.x. *See* OpenWindows, X Window System.

Supercache: 1-MB cache memory on some SPARCstation 10 and SPARCstation 20 models (10/41, 10/51, 20/51, 20/61, 20/71). *See* cache memory.

supermini: Former, "extra powerful" minicomputers, such as VAX 11/7xx and VAX 8xxx, used in scientific environments.

SuperSPARC: SPARC-based CPU chip architecture with outstanding performance due to a special architecture. The SuperSPARC performs three instructions in parallel and has 3 million transistor functions, including 3-KB on-chip cache memory. This results in excellent performance figures: 45 SPECint92 and 54 SPECfp92 at 36-MHz clock rate, and 126 SPECint92 and 121 SPECfp92 at 75 MHz (with 1-MB external cache memory).

superuser: Special user who is allowed on a system to read any file from any owner, to write into any file, and to execute any executable file. With UNIX, the superuser normally has the username `root`. *See* permission, root.

surface analysis: Procedure usually performed after formatting a disk for detecting bad tracks.

SVR4 (System V, release 4): *See* System V.

swapping: Different from paging, swapping is the transfer of entire processes from RAM into the swap space. To free up memory, UNIX decides to swap out processes that have been idle for a certain time, usually a matter of seconds or minutes, or that have a low priority. When such a process is reactivated, perhaps by clicking on a window that has not been used for a while, there should only be a slight delay of a second or so until the process is ready again. The delay might be because another process has to be swapped out in turn, but mainly it is because the data have to be read into RAM again. Swapping and paging can be avoided by adding more RAM. *See* swap space, paging.

swap space: UNIX uses a special disk partition as extension of the RAM memory. The ordinary UNIX file systems are complex and, therefore, not optimized for speed of disk transfers. The swap space, on the other hand, is organized differently, so that every time the active programs request more memory than currently available, UNIX quickly moves inactive processes to the swap space on disk. In addition, if a single process requires more memory than the amount of memory physically installed, UNIX can use virtual memory capabilities to load and off-load data in pages of 8 KB or 4 KB from and to the swap space. *See* paging.

switch: *See* power switch.

SX: Sun 24-bit graphics accelerator used in some SPARCstation 20 models (20/71SX, 20/151SX). The SX provides 24-bit color (16,777,216 colors) and is optimized for fast windowing, imaging, and fast video playback. The SX graphics accelerator is not compatible with Solaris 1.x. *See* SPARCstation 20/71SX, SPARCstation 20/151SX.

symbolic link: An alias or alternate name for a file in the UNIX file system. A symbolic link is a type of pointer to a file in another directory, which can help simplify paths. For example, `/export/home/vnmr` is accessible as `/vnmr` through a symbolic link `vnmr` in the `/` directory, pointing to `/usr/vnmr`. Unlike hard links, symbolic links can be made across partition limits. When such a file is deleted, the symbolic links are not removed. On the other hand, when a file is removed is a symbolic link, only the link is removed, not the file itself. *See* hard link.

synchronous: Transmission of data over dedicated lines with synchronization signals that, in essence, determine the speed of communication. Synchronous transmission is more demanding of the hardware than asynchronous I/O, but is potentially faster, with up to 64,000 baud on the RS-232 interface. In the RS-232 standard, the lines 15, 17 and 24 for synchronization are used in addition to some of the normal handshaking lines 4, 5, 6, 8, and 20. The SBus in desktop SPARCstations uses synchronous data transfers for optimum reliability and speed. *See* asynchronous, handshaking.

System V: The current version of UNIX from the AT&T Bell Labs. System V UNIX was developed in 1984 from System III, which originated indirectly from UNIX Version 7, a version written for the PDP/11 computer. SunOS 4.x combines 4.2BSD UNIX with the most important features of release 3 of System V. System V.3 is more popular, but has some specific disadvantages over 4.2BSD, such as a file system organization that is slower and suffers from aging, getting slower and slower with the time of use. Periodic cleanup operations are necessary to maintain acceptable performance. Together with AT&T, Sun has developed System V, release 4 (often abbreviated as SVR4). This is now marketed as Solaris 2 and has become the standard Sun operating system. SVR4 combines the advantages of 4.2BSD UNIX with those of System V.4. Sun-3 and earlier systems are not compatible with Solaris 2 and can therefore not be upgraded to this level of operating system. *See* UNIX system, SunOS, Solaris 2.

T-switch: Manual switch that allows using two I/O devices alternatively, but not simultaneously. If the two devices talk a different language, such as a plotter and a printer, the user must be careful to completely terminate the communication with the device that has to be disconnected. A terminal and an output device, such as a printer, must not be connected to the same port via a T-switch, because the handshaking signals of the output device would cause the port to initiate a login procedure, which would then interfere with any attempt to send data to the plotter.

tape: Magnetic medium to store digital information, mostly for backup purposes or for transporting large amounts of data. Current Sun workstations support 1/4-inch tapes (150 MB and 2.5 GB), DAT tapes (5 GB), and 8 mm Exabyte tape drives (2.5, 5, 10, and 14 GB). Early versions of the Sun operating system were delivered on QIC-24 tapes. All Sun-3 systems were equipped with a QIC-24 (60 MB) drive. Many SPARCstations are equipped with the QIC-150 tape drive, which can also read QIC-24 and QIC-11 data. Software from Sun and Varian is now all delivered on CD-ROM. This means that tape drives are used mostly today for backup and mass storage purposes. *See* QIC, Exabyte, DAT.

tape archive: *See* tar.

tape cassette: Cassette with 1/4-inch tape. Recommended quality is 3M DC600A (600 feet), or equivalent, for maximum (60 MB) storage capacity. For SPARCstations with QIC drive, 3M DC6150 (600 feet, 150 MB), or equivalent, must be used. Other tape formats also come in cassettes: 8-mm 5-GB Exabyte tapes and DAT cartridges. *See* QIC, Exabyte, DAT.

tape drive: Device for reading and writing magnetic tapes. Most Sun systems are equipped with one or two tape drives. The drives come in various formats and can also be used remotely. *See* tapes, QIC, Exabyte, DAT.

tap kit: Device that connects a transceiver to an Ethernet cable. Three different kinds of tap kits are in use: thin Ethernet tap kits that connect via BNC-type T-piece, N-type tap kits that connect to thick Ethernet cable via N-type connectors, and vampire tap kits that connect onto standard thick Ethernet cable without disrupting the network. Mounting a vampire tap kit onto Ethernet coaxial cable requires special tools and careful operation because the cable can be inadvertently shorted. With the newer twisted-pair Ethernet, standard tap kits are no longer required. *See* Ethernet, twisted pair.

tar: UNIX tape archive program for reading and writing multiple files stored on a magnetic tape. Useful also for transferring data between different file systems.

task: Single execution of a program. Under UNIX, the term “process” is normally used instead of task, although both terms mean the same thing.

Tcl/Tk (Tool command language /Tool kit): Tcl/Tk (pronounced “tickle tee-kay”) is an application programming interface (API) used in the development of interactive applications or graphical user interface (GUI) under UNIX. Tcl is a simple interpreted scripting language, similar to a UNIX shell or a macro language. Tk is a toolkit that provides facilities, such as buttons, menus, and event handlers, for creating an interactive GUI. Starting with version 5.2, VNMR began using Tcl/Tk for specific interactive utilities, such as a spinner control tool, the `enter` program for automation, and various configuration and administration tools. Sun and Netscape have announced a Tcl/Tk plug-in for Netscape Navigator that will allow calling HTML pages with Tcl/Tk scripts and will facilitate the development of interactive Web pages. *See* API, Netscape Navigator, HTML.

TCP (transport control protocol): Networking transport protocol that uses the IP network protocol for the transfer of information over Internet. TCP provides error-free, in-order, and two-way data IP packet transfer. TCP is used by higher-level services, such as `rcp`, `rlogin`, `rsh`, and `ftp`. TCP is often named in connection with IP (TCP/IP), even though the two protocols are two totally different entities and IP is used by other services, such as UDP. *See* IP, TCP/IP, Internet, UDP.

TCP/IP (**transport control protocol/internet protocol)**: Networking protocols for communication and data transfer via Ethernet. Available on all Sun systems, most UNIX systems, and many PC compatibles and Apple Macintoshes. *See* Ethernet, NFS.

terabyte (TB): 1,099,511,627,776 bytes, which corresponds to 1024 gigabytes (GB).

terminal: Operator interface consisting of at least a screen and a keyboard.

terminator: Device used on buses and open rf cables to avoid signal reflections and erroneous signal transmission. *See* bus terminator, Ethernet terminator.

text file: File that contains ASCII characters only and can be accessed by text editors, such as `vi` and `textedit`, as well as many UNIX commands, including `cat`, `more`, and `nroff`. Each character in a text file occupies 1 byte in memory or on the disk. VNMR was written to work with text files as much as possible, because such files can be accessed, modified, and inspected easily, as opposed to binary files, which are normally only accessed by special commands or dedicated programs. *See* binary file, directory file.

text processing: Off-line text formatting program, as opposed to word processors. With text processors, the formatting instructions are inserted into the text in the form of special character sequences and whenever the formatted text is requested, the text processing scans through the entire text and interprets the formatting instructions, producing the formatted document as output on a suitable printer or on the screen. Standard UNIX software includes the text formatting programs `nroff` and `troff`.

textedit: An easy-to-use mouse- and window-oriented UNIX text editor. Textedit is always in the insert mode and has features such as drag-and-drop and window splitting, which allow working on various portions of the text simultaneously. Experienced UNIX users probably prefer the `vi` text editor, which has a less sophisticated user interface but is faster and more powerful, especially for repetitive tasks. *See* vi, text file.

TGX (TurboGX): Fast single-chip 8-bit color graphics accelerator with fast windowing and 2D/3D wireframe graphics. TGX is the successor of the GX graphics accelerator and is used in the SPARCstation 5, SPARCstation 10, some SPARCstation 20 (20/50, 20/51, 20/61), and in some Ultra 1 (1/140, 1/170) workstations.*See* GX.

thin Ethernet: Ethernet based on 50-ohm BNC cable connections. The cables must have terminators on both ends and no branching is allowed. *See* Ethernet.

TIFF (**tagged image file format):** Popular format for storing graphics. First created by Aldus Corp., it was later adopted by most PC and Macintosh painting, imaging, and desktop publishing programs. TIFF can be used to store both grayscale and color graphics. On PCs, TIFF files usually have a .tif extension. *See* GIF, JPEG.

time sharing: An operating system that shares CPU time between several tasks, or processes, running simultaneously. This involves priority handling and interrupting long processes as frequently as every second to let other processes have their share of CPU time. All this is completely transparent to the user, and if only one task is mainly active, there is be no significant slow-down in performance as compared to a system without time sharing. *See* process, task, multitasking system.

tool: Program that can be used to build complex programs. Most UNIX programs can be regarded as tools for building powerful shell scripts, while MAGICAL commands are tools for building macros. A shell tool within OpenLook is just a window with a command interpreter for performing any UNIX task.

top-down programming: An approach in programming that describes the sequence or the timing in which the various parts of a program are written. First, a very general main program is written, for example:

```
#include <stdio.h>
main()
{
   getdata();
   calculate();
   showresult();
}
```

Then, the subroutines, such as `getdata()`, `calculate()`, and `showresult()` in the example, are defined, details are filled in for defining further subroutines, and so on, down to the smallest details. *See* structured programming.

transceiver: On Ethernet networks, a combined transmitter and receiver that converts serial binary information into modulated 10 MHz rf and vice versa. The transceiver is directly hooked up to the Ethernet cable via a tap kit. Branching from the Ethernet cable to the transceiver is not allowed. The transceiver is not a terminator.

transfer rate: Speed at which data are transferred continuously over an interface, or to and from a hard disk. SCSI disks transfer data at 2 to 4 MB per second. The SCSI-2 bus in the SPARCstation 10 and 20 operates at up to 10 MB per second, the fast/wide SCSI in Sun Ultra systems with Creator graphics supports transfer rates of up to 20 MB per second. The transfer rate depends on a number of factors: bus speed, average seek time to access tracks, latency and interleaving, and amount of fragmentation in the file or in free space when writing a file. To minimize track-to-track access time and increase file transfer rates, the fast BSD 4.2 file system used in Sun workstations tries to keep files in subsequent tracks, or cylinder groups. *See* access time, latency, interleave factor, cylinder group.

tree structure: Hierarchic structure of a directory file with subdirectories, subdirectories of subdirectories, and so forth, and their subfiles. To avoids loops, a proper tree structure does not allow for two links to the same subfile. *See* directory file, link

troubleshooting: Activities such as rebooting the software, checking electrical connections, and consulting manuals or course notes. If nothing helps, call service for hardware problems, an applications chemist for software problems, or your salesperson and your boss for monetary problems, if additional hardware is required.

TurboSPARC: SPARC CPU chip manufactured by Fujitsu that is fully compatible with the MicroSPARC II CPU, but operates at higher clock rates (170 MHz instead of 110 MHz), offers more primary cache (16 KB as opposed to 6 KB) and secondary cache (512 KB), and can be used in the Sun SPARCstation 5/170. The current implementation runs at 170 MHz and performs at 2.91 SPECfp95 and 3.32 SPECint95. This chip boosts the performance of the SPARCstation 5 to the level of a SPARCstation 20. *See also* MicroSPARC, SPARC.

twisted-pair Ethernet: Ethernet standard that allows Ethernet communication over telephone cable-like hardware (unshielded twisted pair, UTP). Newer SPARCstation and Ultra workstations come with a twisted-pair Ethernet connector, besides a standard Ethernet (transceiver cable) connector. Unlike traditional Ethernet, which always requires a linear backbone without branching, twisted-pair Ethernet also allows for star-shaped network topologies. *See also* UTP.

UC Berkeley: *See* 4.2BSD.

UDP (user datagram protocol): A simple data transport protocol that builds on the IP network layer and protocol. UDP is faster but less robust than TCP and is used for services such as HTTP, where information is transported primarily for visual presentation and data integrity and error-checking are less important than with other services, such as `ftp` and `rcp`. *See also* IP, TCP, Internet, HTTP.

UI: *See* Unix International.

Ultra 1: Family of single-processor UltraSPARC desktop workstations with 3 SBus slots, 32 to 1024 MB of ECC RAM, up to 2 internal hard disks, optional internal floppy and CD-ROM drive, UPA system interconnect, and TGX or Creator/Creator3D graphics. *See* UltraSPARC, TGX, Creator, UPA, SBus, ECC.

Ultra 1/140: Ultra 1 workstation with 143-MHz UltraSPARC CPU, 512-KB secondary cache, and TGX graphics accelerator (215 SPECint92, 303 SPECfp92; 5.41 SPECint95, 7.90 SPECfp95). *See* Ultra 1, UltraSPARC, TGX.

Ultra 1/140E: Same as Ultra 1/140, but with Creator or Creator3D graphics accelerator, 1 UPA port, 2 Sbus slots, fast/wide SCSI (SCSI-3) interface, and fast Ethernet (100BaseT). *See* Ultra 1, Ultra 1/140, UltraSPARC, Creator.

Ultra 1/170: Ultra 1 workstation with 167-MHz UltraSPARC CPU, 512-KB secondary cache, and TGX graphics accelerator (252 SPECint92, 351 SPECfp92; 6.26 SPECint95, 9.06 SPECfp95). *See* Ultra 1, UltraSPARC, TGX.

Ultra 1/170E: Same as Ultra 1/170, but with Creator or Creator3D graphics accelerator, 1 UPA port, 2 Sbus slots, fast/wide SCSI (SCSI-3) interface, and fast Ethernet (100BaseT). *See* Ultra 1, Ultra 1/170, UltraSPARC, Creator.

Ultra 1/200E: Same as Ultra 1/170E, but with 200-MHz UltraSPARC CPU (7.44 SPECint95, 10.4 SPECfp95). *See* Ultra 1, Ultra 1/170E, UltraSPARC.

Ultra 10: Single-processor UltraSPARC desktop workstation with one 300-MHz UltraSPARC IIi CPU (12.1 SPECint95, 12.9 SPECfp95), equipped with 4 PCI bus slots (33 MHz, 32 bits), 64 to 1024 MB of ECC RAM, 0.5-MB secondary cache, 1 or 2 internal 4.3-GB EIDE hard disks, optional internal floppy and CD-ROM drive, UPA system interconnect, PGX or high-end (Creator, Creator3D, or Elite3D) graphics. *See* UltraSPARC, EIDE, PGX, UPA, PCI bus, ECC.

Ultra 2: Family of single- or dual-processor UltraSPARC desktop workstations. 64 to 2048 MBytes of ECC RAM, 1 MByte secondary cache per processor, 4 SBus slots, fast/wide SCSI (SCSI-3) interface, up to 2 internal hard disks, optional internal floppy and CD-ROM drive, UPA system interconnect, 1 UPA port, Creator or Creator3D graphics, fast Ethernet (100BaseT). *See* UltraSPARC, Creator, UPA, SBus, ECC.

Ultra 2/1170: Ultra 2 workstation with one 167-MHz UltraSPARC CPU (6.34 SPECint95, 9.33 SPECfp95). *See* Ultra 2, UltraSPARC.

Ultra 2/1200: Ultra 2 workstation with one 200-MHz UltraSPARC CPU (332 SPECint92, 505 SPECfp92; 7.72 SPECint95, 11.4 SPECfp95). *See* Ultra 2, UltraSPARC.

Ultra 30: Family of single-processor UltraSPARC desktop workstations in a minitower configuration with 4 PCI bus slots (32/64 bits, one at 66 MHz, 3 at 33 MHz), 128 to 2048 MB of ECC RAM, 2-MB secondary cache, up to 2 internal hard disks, optional internal floppy and CD-ROM drive, UPA system interconnect, and TGX, Creator/Creator3D, or Elite3D graphics. *See* UltraSPARC, TGX, Creator, Elite3D, UPA, PCI bus, ECC.

Ultra 30/250: Ultra 30 workstation with one 250-MHz UltraSPARC CPU (10.0 SPECint95, 14.9 SPECfp95). *See* Ultra 30, UltraSPARC.

Ultra 30/300: Ultra 30 workstation with one 300 MHz UltraSPARC CPU; 12.1 SPECint95, 18.3 SPECfp95. *See* Ultra 30, UltraSPARC.

Ultra 5: Single-processor UltraSPARC desktop workstation with one 270-MHz UltraSPARC IIi CPU (9.1 SPECint95, 10.1 SPECfp95), equipped with 3 PCI bus slots (33 MHz, 32 bits), 64 to 512 MB of ECC RAM, 0.25-MB secondary cache, 1 internal 4.3 GB EIDE hard disk, optional internal floppy and CD-ROM drive, and PGX graphics. *See* UltraSPARC, EIDE, PGX, PCI bus, ECC.

Ultra 60: Single- or dual-processor UltraSPARC desktop workstation with 300-MHz or 360 MHz UltraSPARC II CPUs (300 MHz, 1 CPU: 13.0 SPECint95, 18.3 SPECfp95; 2 CPUs: 13.1 SPECint95, 23.5 SPECfp95; 360 MHz, 1 CPU: 16.1 SPECint95, 23.5 SPECfp95; 2 CPUs: 16.1 SPECint95, 29.5 SPECfp95), with 4 PCI bus slots (32/64 bits, one at 66 MHz, 3 at 33 MHz), 128 to 2048 MB of ECC RAM, 2-MB (300 MHz) or 4-MB (360 MHz) secondary cache, 1 or 2 internal hard disks, optional internal floppy and CD-ROM drives, UPA system interconnect, and TGX, Creator/Creator3D, or Elite3D graphics. *See* UltraSPARC, TGX, Creator, Elite3D, UPA, PCI bus, ECC.

Ultra 60/1300: Ultra 60 workstation with one 300 MHz UltraSPARC CPU (2 MB level 2 cache); 13.0 SPECint95, 18.3 SPECfp95. *See* Ultra 60, UltraSPARC.

Ultra 60/1360: Ultra 60 workstation with one 360 MHz UltraSPARC CPU (4 MB level 2 cache); 16.1 SPECint95, 23.5 SPECfp95. *See* Ultra 60, UltraSPARC.

Ultra 60/2300: Ultra 60 workstation with two 300 MHz UltraSPARC CPUs (2 MB level 2 cache each); 13.1 SPECint95, 23.5 SPECfp95. *See* Ultra 60, UltraSPARC.

Ultra 60/2360: Ultra 60 workstation with two 360 MHz UltraSPARC CPUs (4 MB level 2 cache each); 16.1 SPECint95, 29.5 SPECfp95. *See* Ultra 60, UltraSPARC.

UltraSPARC: First SPARC-based 64-bit high-performance CPU chip architecture from Sun. The UltraSPARC performs several instructions in parallel (in a 9-stage pipeline, 4 instructions per cycle) and includes 32-KB onchip cache memory (supporting up to 4 MB of external, secondary cache). The UltraSPARC does memory I/O at up to 1.3 GB per second on a 128-bit I/O path. The instruction set has been expanded by a special visual instruction set (VIS) that dramatically accelerates graphics and imaging. The UltraSPARC is one of the most powerful microprocessors on the market and performs at 5.41 SPECint 95 (215 SPECint92) and 7.9 SPECfp95 (303 SPECfp92) at 143-MHz clock rate, and up to 7.72 SPECint 95 (332 SPECint92) and 11.4 SPECfp95 (505 SPECfp92) at 200-MHz. In parts, this performance is made possible by a new system interconnect architecture, the UPA (UltraSPARC Port Architecture), which removes bottlenecks in the communication between the CPU and other computer components. *See* SPARC, Ultra 1, Ultra 2, UPA, VIS.

UltraSPARC II: The second generation SPARC-based 64-bit high-performance CPU chip architecture from Sun. The UltraSPARC II performs at 10.0 SPECint 95 and 14.9 SPECfp95 at 250-MHz clock rate, up to 12.1 SPECint 95 and 18.3 SPECfp95 at 300-MHz, and up to 16.1 SPECint 95 and 23.5 SPECfp95 at 360-MHz. *See* SPARC, UltraSPARC, Ultra 30, Ultra 60.

UltraSPARC IIi: Variant of the UltraSPARC II CPU chip that also includes a PCI bus controller, used in the Ultra 5 and Ultra 10 workstations. Currently available at 270 MHz (Ultra 5, 9.1 SPECint95, 10.1 SPECfp95) and 300 MHz (Ultra 10, 12.1 SPECint95, 12.9 SPECfp95). *See* UltraSPARC II, UltraSPARC, Ultra 5, Ultra 10.

UniPack: Small desktop enclosure for Sun peripherals, containing one or several disks, a tape drive, or a CD-ROM drive UniPack replaces the desktop storage pack used since the introduction of the SPARCstation 1. *See* desktop storage pack.

UNIX International (UI): Non-profit organization originally founded by Sun Microsystems and AT&T to create and promote UNIX System V, release 4 (SVR4) and associated software. Over 150 other companies have joined Sun and AT&T in UI. Fearing a monopolization of the UNIX market, a number of other companies created the Open Systems Foundation (OSF), which also created a UNIX standard. *See* System V.

UNIX system: Operating system invented by Ken Thompson and Dennis Ritchie at AT&T Bell Laboratories for the PDP-7 minicomputer. Initially written in assembly language, Version 3 UNIX was translated into C in 1973. Distribution to the public outside Bell Labs started in 1974 with Versions 5 and 6. Version 7 in 1979 was the basis for all of the following UNIX implementations. Since then, UNIX has split up into a family of operating systems such as XENIX and HP-UX. AT&T developed System III in 1982 and System V in 1984. System V is probably the most popular version today. In 1980, the U.S. Department of Defense chartered the University of California in Berkeley to redesign UNIX for distributed computing on networks. The results of this cooperation were 4.1BSD and 4.2BSD (Fourth Berkeley Software Distribution), which featured a faster and better file system, network support, largely expanded process address space, and new utility programs such as a fullscreen editor (`vi`) and an alternate command interpreter (C shell). 4.2BSD became the starting point for SunOS. *See* System V.

UPA (UltraSPARC Port Architecture): New system interconnect architecture used in Ultra 1 and Ultra 2 workstations. UPA is a cross-bar switch, a concept used only in supercomputers so far. It implements packet-switched, multiway communication at very high throughput (1.3 GB per second) between the CPU (144-bit path), the memory (288-bit path on Ultra 1; 576-bit path on Ultra 2), the networking and SBus I/O unit (72-bit path), and the graphics processor. UPA allows for simultaneous, parallel data transfers between different units. All data paths include ECC (error checking and correction) through parity bits (one parity bit per 8 bits path width). *See* UltraSPARC, ECC.

upgrading: Most Sun workstations can easily be upgraded with additional or more recent hardware.Check with your Varian salesperson. Of course, you can purchase hardware upgrades from Sun or other vendors but, in that case, Varian cannot take responsibility for the additional hardware. Devices from third party manufacturers may not be fully compatible, especially on a spectrometer, and you may have additional problems if this upgrade affects the performance of the standard hardware, so check first with your Varian salesperson or a Varian specialist.

URL (uniform resource locator): Hypertext address for a remote resource in HTML. The principal format of a URL is as follows:

`protocol``://`*`user`*`:`*`password`*`@`*`host.domain`*`:`*`port/path/file#anchor`*

where *`protocol`* is most often `http` (for HTTP) or `ftp` (for FTP), *`user`* and *`password`* are the user name and password for non-anonymous FTP, *`host.domain`* is an Internet host address (such as `www.nmr.varian.com`), *`port`* specifies a non-standard port in cases where such port addresses are used (e.g., where a server has both a public and a private port or interface for the same protocol), *`path`* is the directory containing the resource (if *`path`* is not specified, the user to led into the default directory), *`file`* is the local file path to the resource (if *`file`* is not specified, the user is shown a directory listing, if FTP, or a document named `index.html` or `index.htm` is transferred, if HTTP), and *`anchor`* (HTTP only) is a tag or label within an HTML document that the user jumps to when the document is displayed. Some examples:

```
http://www.varian.com
http://www.nmr.varian.com/products/software/
http://www.nmr.varian.com/news/events.html
ftp://prep.ai.mit.edu/u2/emacs/
ftp://usergroup@vnmrnews.nmr.varian.com/pub/userlib/README.FIRST
```

See also HTML, WWW.

user: Person that has direct access to the UNIX software. For each new user, the superuser `root` installs the necessary files in the `/home` directory and starts an account for the user. For safety reasons, `root` should assign an initial password for new users. Users other than `root` have only limited write access to files outside their home directory. Each user has a home directory in `/home`. *See* superuser, root.

user partition: General-purpose partition for users, typically with home directories. On the Sun host, the `/usr` directory normally contains only software and global data files for UNIX. The home directories and VNMR are stored in the `/export/home` directory on a separate disk partition of the main disk.

utility: UNIX programs—including editors, command interpreters, and compilers—that run on top of the kernel. Basically, any UNIX program that is not a kernel function can be called a utility.

UTP (unshielded twisted pair): Telephone cable-like cabling used for 10baseT (standard) and 100baseT (fast) Ethernet, and for ATM. Transfer rates can be as high as 155 Mbps. *See also* Ethernet, ATM, STP.

vi: Powerful and frequently used fullscreen UNIX text editor. `vi` (pronounced "vee-eye") is compatible not only with the Sun graphics interface, but also with RS-232 terminals, because it does not rely on the mouse. The cursor is moved around with cursor keys or with special commands instead. `vi` is powerful, but definitely more complex than `textedit`, the other major UNIX text editor. `vi` operates in three modes: insert mode, `vi` command mode, and last line or `ex` command mode (`ex` is a line-oriented text editor often used for global changes). As the standard text editor on the Sun host, `vi` is called by many VNMR commands. *See* textedit, text file.

virtual memory: By dividing every process into pages of 8 KB or 4 KB, individual pages can be loading and unloading instead of processes. This allows working with a processes up to 1 GB, much more than the actual RAM memory. *See* process.

virus: Generally a small, but sometimes quite dangerous, software bug that can multiply within the software, infecting files—especially executables. A virus can destroy an entire operating system, but fortunately, viruses are mainly an issue in the area of PCs and Macintoshes, and UNIX has not so far suffered from viruses.

VIS (Visual Instruction Set): The UltraSPARC instruction set has been complemented by a special visual instruction set (VIS) that dramatically accelerates graphics and imaging. Pixel manipulations that otherwise would take many steps can be executed in a single clock cycle. VIS also accelerates image processing, real-time video compression, such as JPEG and MPEG, and image animation. *See also* UltraSPARC, MPEG, JPEG.

VME: Industry standard bus structure. Implemented as part of the backplane, VME is the main expansion bus on Sun 3/xxx and 4/xxx computers. As used in Sun computers, the VME bus is 32 bits wide and allows continuous data transfer rates of up to 8 MB per second between different computer boards. The VME bus is also used in the current Varian NMR acquisition computers.

VNMR: Varian's research NMR software package. It was primarily designed for Sun-based VXR and UNITY NMR spectrometers using Sun graphics workstations as the host computer, but has been expanded into other software and hardware platforms. VNMR is currently available for Sun SPARCstation and Ultra workstations running Solaris 2.x (VnmrX), for the IBM RS-6000 Powerstation series (VnmrI), and for Silicon Graphics workstations operating under IRIX 4.0.x, 5.x or 6.x (VnmrSGI). *See* VnmrX, VnmrI, VnmrSGI.

VnmrI: Version of VNMR for the IBM RISC Powerstation 6000 series of graphics workstations. The RS/6000 series workstations are computers with strong floating point performance. IBM computers and VnmrI cannot be used to drive a spectrometer. VnmrI runs under AIX Version 3 and OSF/Motif, IBM's version of X Window System software. *See* VNMR.

VnmrS: Version of VNMR developed for the Sun-3 and Sun-4 (including SPARCstation) series of Sun graphics workstations, running SunView windowing software. This version

of VNMR can be used to drive a Sun-based UNITY*plus*, UNITY or VXR spectrometer. It supports GraphOn black-and-white and Tektronics (4105, 4107, 4205, and 4207) color graphics terminals. Varian has stopped further developments on VnmrS software. VnmrS releases include:
VnmrS 1.1, 1.2: SunOS 3.2 - 3.5: Sun-3 only
VnmrS 2.1, 2.2, 2.3: SunOS 3.5: Sun-3, SunOS 3.2: Sun-4
VnmrS 3.1: SunOS 4.0.3: Sun-3 and Sun-4
VnmrS 3.2A: SunOS 4.0.3, SunOS 4.1: Sun-3 and Sun-4
VnmrS 3.2B: SunOS 4.1.1: Sun 4/xx only, identical to 3.2A otherwise
VnmrS 4.1A: SunOS 4.1.1
VnmrS 4.1B: SunOS 4.1.2 and 4.1.1
VnmrS 4.1C: SunOS 4.1.3, 4.1.2 and 4.1.1
VnmrS 4.2: SunOS 4.1.3, UNITY*plus* only
VnmrS 4.3: SunOS 4.1.x, VXR-S / UNITY / UNITY*plus*
VnmrS 5.1A: SunOS 4.1.3, 4.1.2 and 4.1.1 (not for Sun-3)

VnmrSGI: X Window System version of VNMR for Silicon Graphics computers running IRIX 4.0.x, 5.x, or 6.x. *See* VNMR.

VnmrV: Version of VNMR for VMS-based VAX and MicroVAX computers from DEC, including the MicroVAX II, VAX 11/7x0, and VAX-8xx0. VnmrV does *not* support high-resolution graphics interfaces (those are not available on VMS-based systems), but only terminal operation (GraphOn black-and-white and Tektronics color graphics terminals). The last release was VnmrV 4.1 and VnmrV has not developed any further. See VNMR.

VnmrX: Version of VNMR for the Sun, but it supports the X Window System, not SunView as does VnmrS. Because it is based on X, it also supports X servers as well as GraphOn black-and-white and Tektronics color graphics terminals. VnmrX also supports all acquisition functions and can be used to drive any Sun-based Varian spectrometer. With Solaris 2, SunView is no longer supported, and VnmrX has become the standard version of VNMR for controlling spectrometers. *See* VNMR. VnmrS releases include:
VnmrX 4.1: SunOS 4.1.2 and 4.1.3
VnmrX 4.3A: SunOS 4.1.x
VnmrX 5.1A for SunOS 4.1.x
VnmrX 5.1A for Solaris 2.3, 2.4, 2.5, 2.5.1
VnmrX 5.1B: SunOS 4.1.x, Solaris 2.3, 2.4, 2.5, 2.5.1
VnmrX 5.2F: Solaris 2.3, 2.4, 2.5, 2.5.1 (UNITY *INOVA* and GEMINI 2000 only)
VnmrX 5.3: Solaris 2.4, 2.5, 2.5.1
VnmrX 6.1A: Solaris 2.4, 2.5, 2.5.1, 2.6

VRAM: Fast random-access memory (RAM) chips that are optimized for use as the hardware frame buffer in video controllers: VRAM chips are dual-ported with one port used for updating and changing the display, and the other port is used to read out the display information. *See* RAM, 3D-RAM.

VRML (Virtual Reality Modeling Language): An extension to HTML, introduced by SGI, that enables defining and manipulating 3D objects within hypertext documents across the Web. *See* HTML, SGI, WWW.

W3: *See* WWW.

WABI (Windows Applications Binary Interface): Sun software that emulates the Microsoft Windows 3.1 operating environment under X Windows by translating Microsoft Windows library calls into the corresponding X Windows calls, and emulating x86 calls on the SPARC processor. WABI permits running a large range of PC software packages written for Windows 3.1. The current version (WABI 2.2) supports PC software such as

MS Office, MS Word for Windows, MS PowerPoint, MS Excel, CorelDraw, and WordPerfect.

wait states: CPU cycles that are spent waiting for data to arrive in the CPU registers from the RAM memory. Because RAM memory is considerably slower than the CPU clock rate, often the CPU is idling for a few cycles. Modern CPU architectures try to avoid such wait cycles by using an intermediate, fast-cache memory between the RAM, the hard disk, and the CPU. *See* cache memory.

WAN (wide-area network): A large-scale computer network across several cities or countries

Web: *See* WWW.

Web authoring: The process of creating and designing HTML documents. *See* hypertext, HTML, WWW.

Web authoring tool: Software for creating and designing HTML documents. *See* hypertext, HTML, WWW.

Web browser: Program that requests, receives, formats and displays hypertext (HTML) documents from a Web server, using the HTTP protocol, allows for user interaction with HTML documents, and provides a convenient access to the FTP protocol for downloading text and binary data. The first browsers were pure ASCII text interfaces, but graphical browsers such as Netscape Navigator, HotJava, and Mosaic are most popular on UNIX systems now. These browsers offer true multimedia access to the Web, with on-screen animation and audio support. *See* Mosaic, Netscape, HotJava, Java.

Web page: A HTML document in general or the entry point into a series of HTML documents on a Web server. A large number of companies have now opened up Web pages and do business over the Web, and many Web users have defined personal Web pages. *See also* HTML, WWW.

Web server: Networked machine that provides access to hypertext (HTML) documents via HTTP protocol. In the case of a UNIX system, this is achieved with the HTTP daemon, `httpd`. Web server software is available from companies that also market Web browsers. See also HTML, HTTP, Hypertext, WWW.

Winchester disk: General term (not a trademark) for a sealed hard disk, mostly with several internal disk platters. *See* hard disk.

windows: Dividing the screen into separate areas for different tasks. With windows, the screen is no longer reserved for a single task, such as a command interpreter or a text editor or a graphics program. Each window is clearly limited by a frame and usually has a title bar that indicates the kind of task running in the window. Windows can overlap and can even be completely covered by other windows. It is also possible to close a window but not destroy it. The window is then shown as a small icon. Invariably, windows also imply that some kind of graphics pointer must be used to indicate to the software which window the operator currently wants to work in or what size and location the operator desires for a window. The most popular graphics pointer is the mouse. Because UNIX is a full multitasking and timesharing operating system, all windows shown on the screen are continuously active, even if they are hidden behind other windows or if they are shown as icons. Only one window can have the focus for keyboard entries. This implies that the operator has to move the mouse onto that window and click on it with a mouse button (in VNMR, the left mouse button is used). The keyboard focus is then indicated by a solid frame around the window. In OpenLook (as in X in general), windows are also allowed to move outside the display area. With SunView, this was not possible. *See* icon.

word: The number of bits handled by the computer at a time: 16, 32, or more bits.

word processing: Text formatting software, such as Microsoft Word, that displays the formatting online. *See* text processing, WYSIWYG.

working directory: *See* home directory.

workstation: Complete stand-alone computer, usually with a single operator interface.

WORM drive: "Write once, read many" optical disk: optical disks that can be recorded once, but not rewritten. WORM drives are used for archiving purposes in some (industrial) GLP environments, as WORM disks are certified to hold the information for over 10 years, and the information on WORM disks cannot be altered. Even though WORM disks are very reliable storage media with a capacity of up to 1 GByte or more per side, no format standard for WORM disks was ever defined (which lead to a diversity of incompatible formats), and no major manufacturer built such drives. The DVD-R may once replace the WORM drives. *See* CD-ROM, DVD, DVD-R.

WWW (World Wide Web, W3, Web): Widely used terms referring specifically to the wealth of interconnected systems communicating using protocols, mainly HTTP, the hypertext transfer protocol. Basically, the Web is a world-wide collection of hypertext (HTML) documents with references (hypertext links) to each other, forming a single, cross-linked pool of information, a world-wide information resource. The idea of using hypertext to link documents on different computers, or in general in different parts of the world, was first brought up in 1989 by a group of researchers led by Tim Berners-Lee at the CERN (Centre Européen pour la Recherche Nucléaire) in Geneva. This defined the origin of the World Wide Web. The most convenient way of accessing documents on the Web is via Web browser. With the advent of the first graphical browser, Mosaic, in 1993, the Web rapidly gained popularity. The term "World Wide Web" is sometimes incorrectly taken as synonymous to Internet. *See* HTTP, HTML, Web browser, Web server, Internet.

WYSIWYG (what you see is what you get): Word processing software, such as FrameMaker, that formats the text online so that as the operator types, the text and appearance of the document are shown on the screen as they will look like when the document is printed. WYSIWYG software gives the tightest and fastest possible control over the final text format. The disadvantage is that such software requires a faster CPU to display pages at the same speed as a text editor. *See* word processing.

X: *See* X Window System.

X client: Program that delivers X Window System display information for an X server. The X client can run locally or on a remote computer, in which case the display information is transferred via Ethernet. Note that the X terminology for servers and clients is totally different from the rest of the UNIX world. *See* X Window System, X server.

XON/XOFF: Special characters that can be transmitted back over RS-232 lines by the receiving device, to start and stop the data flow. If XON/XOFF is the only handshaking option selected, a three-wire cable using pins 2, 3 and 7 is sufficient. Optionally, a shielded cable with the shield connected to pin 1 on one end only can be used. If a null modem is required, lines 2 and 3 must be crossed. *See* null modem.

x-ray emission: Emission from the CRT display. Sun workstations fulfill FCC and other local regulations and, therefore, should be of no harm to the operator.

X server: System that acts as an X Window System terminal. It can be software running on a Sun, an IBM RS/6000, a Silicon Graphics computer, a DECstation, or other graphics workstation with X software. An X server can obtain its display information from a local X client or from an X client running on a remote machine through Ethernet. Apart from UNIX workstations, X server software is also available for the Macintosh and for IBM PC compatibles equipped with Ethernet. Dedicated X terminals are also available. *See* X client, X Window System, X terminal, VnmrX.

X terminal: Special terminal containing X server software built into its firmware.

Xt (X Toolkit): API (application programming interface) used for creating applications under the X Window System. *See* API, X Window System.

X Window System: Standardized windowing software, adapted by Sun, DEC, IBM, Silicon Graphics, and many other computer companies. MIT, the originator of X, defined how graphics software (X client) communicates with the window driver (X server), which can be local software or a remote X terminal (*see* X terminal). The actual X software has to be generated on top of this definition. Sun propagated its own standard for that graphics surface, called OpenLook. X is currently supported on Sun computers by VnmrX, on IBM RS/6000 Powerstations by VnmrI, and on Silicon Graphics IRIS Indigo and Crimson workstations by VnmrSGI. *See* X terminal, X server, X client, VnmrX.

year 2000 (y2000) compliance: The millennium change complicates life for computer users in two ways. For one, the hardware clock in some early PC models was designed not to last beyond the year 1999 (this is not a problem with Sun workstations, where the hardware clock is designed to last until the year 2037 (well beyond the life cycle of these workstations). The other problem is, that a lot of software uses only two digits for the year, and this will cause a lot of problems in many places, from easy ones (such as files created in 2001 being listed or sorted as older than those created in 1998), up to severe ones, such as confusion created in banking software. Solaris 2.6 is certified as being y2000 compliant; VNMR itself does not deal with dates (other than through Solaris). Therefore, a workstation running VNMR 6.1A under Solaris 2.6 (or later, for any of the two) can be regarded as being y2000 compliant.

yellow pages: *See* NIS

zero latency: On a disk drive, reading an entire track directly into an intermediate buffer instead of taking time to seek and read a requested sector in the track. This helps in the frequent case where adjacent sectors are also going to be read. With zero latency drives, these additional sectors are then already in the intermediate fast RAM buffer. *See* latency.

zero wait-state: If the memory chips are slower than the CPU chip and the CPU wants to read in or write some information, probably it performs some wait-states. This can drastically affect the performance of a CPU. Some workstations (such as SPARCstations and Ultra workstations) use fast primary and secondary cache memory as an intermediate buffer to reduce the number of wait-states to almost zero. *See* wait states, cache memory.

Symbols

! shell command, 138
!! shell command, 138
!= operator, 196
"" null string, 194
"..." notation, 194
# notation, 56, 190, 205, 212
$ notation, 206, 212
% input/output notation, 201
& notation, 132
& operator, 196
& pointer notation, 200
&& operator, 196
'...' notation, 194
(...) parentheses notation, 132
(void) operator, 197
* pointer notation, 200
* wildcard notation, 134
++ notation, 195
. file name notation, 133
.. file name notation, 134
.c file extension, 203
.clockrc file, 187
.cshrc file, 50, 71, 75, 99, 135, 153, 164, 183, 184–185
.dt directory, 51, 101, 115, 186
.dtdirectory file, 71
.dtprofile file, 50, 71
.exrc file, 185
.filemgrrc file, 187
.indent.pro file, 185
.login file, 50, 71, 99, 100, 102, 165, 183, 183–184, 218
.mailrc file, 183, 185
.netrc file, 243, 244
.openwin-init, 71
.openwin-init file, 102, 186
.openwin-menu file, 71, 102, 187
.profile file, 185
.rhosts file, 229, 232
.tar file name extension, 150
.tiprc file, 263
.wastebasket directory, 187
.Xdefaults file, 71, 102, 186, 187
.xinitrc file, 186
.xinitrc shell script, 102
.xlogin file, 186, 253, 254
/ notation, 133
/* ... */ notation, 190
/data slice, 41
/export/home partition, 41
/opt slice, 41, 42
/usr slice, 43
/usr/openwin partition, 42
/var slice, 41, 42
  insufficient space in, 113
/vnmr directory, 49
= assignment operator, 195
== operator, 196
> notation, 132
> prompt, 94
>> notation, 132
? wildcard notation, 134
@ notation, 207
[...] wildcard notation, 134
\0 null character, 194
^ notation, 87
^ shell command, 138
'...' notation, 133
{...} notation, 191, 197
| notation, 133
| operator, 196
|| operator, 196
~ notation, 263
~ operator, 196
~ symbol, 134

Numerics

1/4-inch tape cartridges, 151
10/100baseT Ethernet port, 224
10/100baseT standard, 337
100baseT standard, 337
10baseT standard, 337
16-bit computer, 337
32-bit computer, 337
32-bit format, 180
3D-RAM technology, 337
4.2BSD software, 337
4Dwm environment, 251
4-track mode, 151
64-bit computer, 337
68881 math coprocessor, 127
90 degree pulse width calibrations (PW90), 298
90-degree pulse length, 269
9-track mode, 151

A

a.out file, 203
aa command, 173
abort action, 338
abortallacqs command, 67
absolute paths, 133
absolute-intensity mode, 269
access time, 338
access to files, 48
accounting tool, 76
acoustic coupler, 338
acq directory, 73
acq_errors file, 165
acqi command, 73, 250
acqigroup1–acqigroup7 definition files, 73
acqparms.h file, 194
acqpresent file, 66, 104
Acqproc command, 65
acqproc command, 173
acqproc login name, 67
Acqproc process, 62, 247
Acqproc process family, 123
Acqstat process, 62
acqstatus parameter, 165
acquisition
  daemon, 104
  processes, 61–64
  status codes, 165
acquisition bus trap, 167
acquisition computer, 28

Index

acquisition computer backplane power supply, 170
acquisition computer shutdown, 104
acquisition console hostname, 227
acquisition diagnostics terminal, 91
acquisition lock, 116
acquisition on X Window systems, 250
Acrobat Reader, 338
Acrobat Reader color flashing, 101
Acroread file, 173
active SCSI bus terminator, 170
active window, 338
ADC overflow warning, 165
Add Group window, 78
Add User window, 80
adding
    disk space in /var, 113
    hard disk, 35, 51
    printer ports, 91
address, 338
address in Ethernet network, 225
addxvfonts command, 254
adept program for spectral editing, 272
adm directory, 113
admintool command, 117
Adobe Acrobat Reader, 173
Adobe Systems Incorporated, 338
air line traps, 317
air plugs, 319
AIX operating system, 338
AIX software, 253
alias command, 132, 338
alias generation, 184
alias shell command, 138
ALU (Arithmetic Logical Unit), 339
ampersand notation, 132
analyze command, 53
angled brackets (< or >) notation, 22
anonymous FTP, 339
ANSI C, 339
Answerbook package, 41
antistatic protection, 35
API (Application Programming Interface), 339
app-defaults file, 115, 186, 187, 258
Apple Macintosh computers, 245
applet program, 339
argument accessibility, 210, 216
argument to command, 339
argument types, 131
argument variables to functions, 200
argv variable, 210
arithmetic lines in C shell, 207
arithmetic with variables, 212
ARPANET, 339
array processor, 339
array variables, 194
arrays, 193
artificial shoulders in printout, 86
ASCII (American National Standard Code for Information Interchange), 339
ASIC (application-specific integrated circuit), 339
askbcc variable, 185
askcc variable, 185
ASM-100 sample changer, 328
asphyxiation hazard, 320
assembly language, 339
assigning new users, 49
assignment operator, 195
asterisk wildcard notation, 134
asynchronous serial communication, 90
asynchronous transmission, 340
ATM (Asynchronous Transfer Mode), 340
audio I/O controller, 27
AUI (attachment universal interface) connector, 340
AUI Ethernet transceiver port, 26
auto_home file, 241, 246
auto_master file, 241
automated acceptance test procedure, 284
automated console tests, 296
automated decoupling performance tests, 298
automated tests with shaped rf, 298
automatic gain errors, 167
automatic lock errors, 167
automatic mounting, 238–240
automatic teller machine (ATM) cards caution, 19
automatic type conversion, 196
automatic variables, 194
automation lock, 116
automount directory, 240
automountd daemon, 241
automounter, 240–241, 246
automounting directory, 112
Autoshim menu, 74
AutoTest, 284
    13C 90 degree Pulse Width Calibration, 306
    13C test descriptions, 306
    configuration tab, 285
    configuring tests, 287
    CPMG T2, 308
    creating probe-specific files, 289
    directory structure, 290
    enabling tests, 287
    entering system Iinformation, 285
    experiment details, 299
    gradient tests descriptions, 307
    history tab, 288
    macros, 293
    RF performance test descriptions (nonshaped channels 1 and 2), 299
    sample requirement, 285
    saving data and FID files from previous runs, 289
    selecting test packages, 287
    standard tests, 296
    startup, 285
    step-by-step procedure, 284
    test library tab, 288
    test report tab, 289
awk command, 137, 214
axial gradients, 275
axial shims, 281

B

back straight quotes notation, 133
back up file systems, 149
backbone on Ethernet, 221
background command execution, 132
background job, 139

background lock, 116
background menu, 101
background process, 340
backplane, 340
backslash notation, 22, 131
backup copies, 175
backup files, 340
backup tape, 340
bad track on disk, 340
bandwidth, 341
bar command, 161
basename command, 214
batch mode, 341
batch of data, 341
battery on Sun CPU, 341
baud rate, 86, 262, 341
beeping control, 134
benchmark tests, 29
Berkeley file system, 106, 108
beta test, 341
bg shell command, 139
bin directory, 102, 110, 114, 115, 218, 341
binary file, 341
binary file compression, 180
bit, 341
bit bucket, 367
bit operators, 196, 207
block devices, 105, 235
block of data, 341
block of statements, 197
block size complete, 165
blocking factor, 151, 152, 155, 177, 178
BNC cable, 341
BNC tap kit, 222
BNC T-connector, 223
BNC thin Ethernet connector, 26
board, 341
boiling point of helium and nitrogen, 318
boolean operations, 196
boot blocks, 108
boot monitor command, 94
boot -r command, 36, 51
boot up system, 94
bootptab file, 62
bootup messages, 341
bootup process, 341
bootup_message file, 51
Bourne shell, 50, 341
Bourne shell command interpreter, 212
Bourne shell customization, 185
Bourne shell scripts, 212–218
braces in C, 191
branch cable, 222, 230, 231
branching flow control, 197–198
break statement, 198, 209
broad peak base, 279
BROWSERDIR variable, 206
bs parameter, 165
BSD file system, 106
BSD UNIX commands, 114
buffer memory, 342
bug in software, 342
bus arbiter, 342
bus hardware, 342
bus master, 342
bus terminator, 342
button, 342
byte, 342

C

C compiler, 114, 203
C program structure, 190
C programming bibliography, 333
C programming language, 189–204, 342
C shell, 50, 99, 342
C shell command interpreter, 205
C shell scripts, 185, 205–211
C source file format, 185
C++ programming language, 342
C13 file, 73
C13.par file, 271
cache file system, 108
cache memory, 24, 342
cachefs system, 108
caddy for CD-ROM drive, 343
calendar manager, 101
call by reference arguments, 200
call by value arguments, 200
cancel command, 89
capacity of disks or memory, 343
card cage, 343
case construction, 210, 215
case keyword, 198
case-sensitive, 131, 343
cat command, 137
catman command, 132
cautions defined, 17
cc command, 203
ccs directory, 114
cd command, 135
CDE, 41, 43, 100, 343
    console message files, 113
    copying text, 139
    customization files, 186
    definition files, 186
    dtpad text editor, 141
    graphical user interface, 100
    initialization, 50
    installation, 39
    running vi text editor, 146
    setup files, 115
    shortcuts, 139
    software directory, 114
    style manager, 101
    toolbar, 101
    vi shell calls, 146
    xdm login interface, 183
cdept macro, 272
CD-R disk, 343
cdrom directory, 112
CD-ROM disk, 343
CD-ROM drives, 32
CD-ROM mount point, 112
CD-ROM mounting, 238
Centronics port, 343
CGI (common gateway interface), 343
change directory, 135

change file ownership, 121, 136
change file permissions, 136
change group ownership, 122
change user-ID to become root, 137
char data type, 192
character arrays, 193
character size, 86
check Ethernet connection, 172
checking the system, 163
chgrp command, 122
child process, 344
chmod command, 120, 122, 136, 218
chown command, 121, 136
CISC (complex instruction set computer), 344
class, 344
classes of Internet addresses, 225
cleaning the system, 163
clear command, 184
clearing the screen, 184
clicking the mouse, 344
client system, 344
clients, 245
clock, 101, 344
clock command, 187
clock frequency, 344
cmdtool command, 103
coaxial cable, 344
codeint data type, 193
Codonics dye sublimation printer, 244
collision between devices, 344
collision of packets, 230
color flashing problem, 101, 173
color names, 188
comma as command separator, 131
command, 344
command grouping, 132
command interpreter, 344
command line, 344
command line interface, 102
command mode, 345
command mode of vi, 142
command output redirection, 133
command path, 102
command separator, 131
command substitution, 133
command tool, 103
comment lines, 190, 205
Common Desktop Environment. See CDE
compiled programs on remote disks, 237
compiler program, 345
compress a file, 138
compress command, 138, 150, 180, 181
compressfid command, 180
compressing data, 150
compressing files, 180
computer, 345
conditional execution, 208, 214
conditional looping, 209, 215
conditional preprocessor statements, 192
config command, 49
config macro, 72, 73, 173
configuration, 345
connecting Ethernet hardware, 222
connection timed out, 171
connector hardware, 345
conpar file, 49, 72
console, 345
console hardware configuration parameters, 172
console messages, 113
console reset switch, 67
console window, 102
constants, 191
constructor, 345
controller ID, 105
conventions used in manual, 22
converged UNIX, 345
coprocessor chip, 345
copy directories between file systems, 153
copy file on a remote host, 232
copy files, 135
core dumps, 164
core files, 164
core memory, 345
coupling constant, 270
cp command, 129, 135, 152, 153, 184
cpio command, 161
CPU (central processing unit), 345
CPU host name, 138
CPU module, 24
crash from bug, 345
creating new directory, 135
creating shell scripts, 218
Creator graphics accelerator, 345
Creator graphics controller, 26
Creator3D graphics accelerator, 345
credit cards caution, 19
cron command, 164, 175, 179
cron file, 113
crontab command, 164
crontab file, 164, 179
CRT display, 345
csh command, 99
Ctrl-c key combination, 138
Ctrl-d key combination, 103, 134, 135, 138, 184, 239, 262, 263
Ctrl-q key combination, 139
Ctrl-s key combination, 138
Ctrl-u key combination, 135, 253
Ctrl-z key combination, 139
current experiment, 116
current processes, 138
current user name, 138
currently defined aliases, 138
cursor on screen, 346
customization files, 115, 183
customized files, 178
cylinder groups, 346
cylinder on disk, 346
cylinder tracks, 108

D

d1 parameter, 172, 269
DAC values, 275
daemon program, 346
damaged root partition, 178
DAT (digital audio tape) drives, 33, 346
data, 346

data compression, 179
data directory, 55, 117, 266
data storage disk slices, 117
data storage options, 29
data transfer testing on Ethernet, 230
data types, 192
date command, 138
DC-6150 tapes, 151, 177
DC-9250 tapes, 151, 177
dd command, 149, 154, 155
debugging, 346
DEC VT-100 terminal definition, 146
DECNET for the Sun, 245
decomp command, 156
decoupler
  90-degree pulse width tests, 271
  field strength, 270, 271
  field strength check, 317
  field strength tests, 271
  frequency from nucleus tables, 72
  power level, 270
  pulse width calibration, 273, 274
  pulses, 270
  rf field homogeneity, 274
  X pulse widths, 274
decrement operators, 195
dedicated phone line, 261, 262
default input, 346
default interactive shell, 205
default operating mode, 96
defaultdomain file, 233
defaultrouter file, 233, 247
defaultsedit command, 185
defect command, 53
defect list for a disk, 53
define statement, 192
degaussing a tape, 151
degaussing the screen, 163, 346
deleting files, 346
density of helium and nitrogen, 319
dept macro, 272
DesignJet plotter, 346
DeskJet printer, 346
deskset utilities, 101
deskside workstation, 347
desktop storage pack, 347
desktop switching, 101
desktop utilities, 101
desktop workstation, 347
destructor, 347
detune the probe, 270
dev directory, 105, 110
developer software, 114
Developer’s Support option, 40
device driver, 347
devicenames file, 49, 85
devices directory, 110
devicetable file, 86
dewar servicing, 323
df command, 109, 137, 163, 265
df file prefix for mail, 264
dfs directory, 111
dfstab file, 236, 246
dhp/dlp parameter, 270
diagnostic software, 347
dialog box, 347
dial-up connections, 261, 262
diff command, 137
differences between files, 137
differential box power down, 66
dioxane sample, 270
dir command, 51
direct data transfer, 262
directory commands, 135
directory files, 126, 347
directory information, 119
directory levels, 133
directory owner, 121
disabling Ethernet, 231
disk address, 105
disk data storage media, 347
disk drives, 29, 347
disk file errors, 167
disk free percentage, 137
disk full problems with mail, 265
disk ID, 106
disk label, 108
disk layout, 40, 107
disk partitions, 105, 347
disk platters, 107
disk slice, 347
disk space, 109, 347
  adding in /var, 113
  usage for mail, 265
disk synchronization, 138
diskless clients, 248
DiskSuite software, 107, 125
display, 348
  aliases, 138
  background jobs, 139
  command lines, 138
  CPU host name, 138
  current user name, 138
  date and time, 138
  disk free percentage, 137
  disk information, 138
  file names, 134
  group ID, 138
  process-IDs, 138
  size of file, 137
  system information, 138
  text from text files, 137
  text string, 137
  user ID, 138
  user name, 138
  users logged in, 138
  working directory, 135
  xterm window, 256
DISPLAY environment variable, 102
DISPLAY variable, 183
dmesg command, 36, 37
dmf parameter, 270
dn parameter, 72
DNS (Domain Name Service), 348
do loop, 198
documentation for Sun and UNIX, 329
dollar-sign notation, 206, 212
done codes, 165

Index

dos2unix command, 158
dot files, 50, 133, 183, 348
dot matrix printer, 348
dots per inch (dpi), 348
double data type, 192
double straight quotation marks notation, 131
dpi (dots per inch), 348
dpp0 port, 86
dps command, 172
dpwr parameter, 270
DRAM (dynamic RAM), 348
ds command, 172
dsk directory, 105
dt directory, 114
dtlogin graphical login, 99
DTP (desktop publishing), 348
dtpad command, 141
dtpad text editor, 101
dtterm shell tool, 101
dtterm shell window, 102
du command, 137, 152, 164
dumb mode terminal emulation, 249
dumb terminal, 184
dump command, 155, 164
dump command (UNIX), 317
dump schemes, 177
DVD (digital versatile disk), 348
DVD-R (DVD-Recordable), 348
DVD-RAM, 348
DVD-ROM, 348

E

EBus (external bus), 349
EBus controller, 26
EBus interface, 27
ECC (error checking and correction) memory, 25, 349
echo command, 137, 262
ed text editor, 147
edit text file, 137
edit text stream, 137
editor program, 349
EDITOR variable, 164, 179
EEPROM (electrically erasable programmable ROM), 349
EEPROM settings, 94
EIDE (enhanced IDE), 349
EIDE disk controller, 27
eject command, 156, 158
ejecting a floppy disk, 158
electronic mail (e-mail), 349
Elite3D graphics accelerator, 349
Elite3D graphics controllers, 26
else statement, 192, 197, 209
emacs text editor, 147
e-mail (electronic mail), 349
e-mail setup, 264
emergency UNIX shutdown, 69
emulation software, 349
encrypted password, 99
end statement, 209
End User Support option, 39
endif statement, 192
endless loops, 199
end-of-file mark, 151
end-of-text (EOT) character, 262
enter program, 219
environment variables, 206, 212
EOT (end-of-text) character, 262
EPROM (erasable programmable ROM), 349
ergonomics, 349
errmsg file, 203
error codes, 165
error during acquisition, 165
error message, 349
error messages for Ethernet, 230
Esc key, 134
etc directory, 111
Ethernet address, 350
Ethernet controller, 26
Ethernet controller boards, 171, 247
Ethernet hardware, 349
Ethernet interface board, 29
Ethernet interface chip, 228
Ethernet node, 222
Ethernet software, 225
Ethernet structure, 221–222
Ethernet terminator, 350
ethers file, 62, 171
EUnet national branches, 227
ex command mode, 144
ex text editor, 144, 147, 185
Exabyte tape compression, 180
Exabyte tape drives, 33, 34, 350
exec command, 187
execkillacqproc script, 67, 123
executable program, 350
execute command on a remote host, 232
execute permission, 119
execution in separate shell, 132
exit command, 103, 251
exit statement, 210, 216
exiting the computing environment, 350
exiting VNMR, 103
exp1 to exp9, 116
expand command, 87
expansion board, 350
expansion capabilities, 29
expansion memory, 350
expansion slot, 350
experiment locking, 116
export command, 212
export directory, 111
expr command, 212
external bus (EBus), 27
external hard disk installation, 35, 36
external swap space, 56
external variables, 193

F

face mask for handling LHe, 320
fast Ethernet, 224, 350
fastboot command, 96
fasthalt command, 96
fb shell command, 139
fdformat command, 156, 157

Federated Naming Service, 112
fiber-optic cable, 350
FID complete, 165
FID file compression, 180
FIFO underflow error, 167
file, 351
file access, 74
file access mode, 351
file compression, 180
file editors customization, 185
file handling commands, 135
file information, 119
file length, 119
file manager, 101
file name rules, 133
file name suffix, 133
file owner, 119, 121
file sizes, 137
file system, 351
file system backup, 149
file system checks, 95
file system mounting, 234
file system overhead, 109
file system scan, 175
file system sharing administration, 111
file system tree, 124
file transfer protocol (FTP), 353
file type, 119
filec option, 134–135
file-creation permissions mask, 74
filemgr command, 187
filter program, 351
find command, 136, 239
find files, 136
finger command, 50
firewall, 351
firewall software, 234
firmware, 351
fixpsg command, 190
flag strings length, 191
flammable gases warning, 18
flickering on CRT display, 351
float data type, 192
floating point calculation, 24
floating point coprocessor chip, 351
floating point number, 192, 351
floating point operation, 351
floating point unit (FPU), 352
floppy directory, 112
floppy disk, 351
floppy disk controller, 27
floppy disk drives, 31, 156
floppy disk mount point, 112
floppy disk mounting, 235
flow controls in Bourne shell, 213
flow controls in C shell, 207
flow rates of nitrogen and helium, 317
flowmeter valve, 323
flush command, 116
flutter tube, 320
fmt command, 89, 137
fn parameter, 172
fold command, 87, 137
font, 351
fonts for IBM RS/6000, 253
fonts for screen display, 71
fonts under OpenWindows, 188
footprint, 352
for loop, 198
for statement, 215
foreach statement, 209
foreground command, 138
foreground lock, 116
foreground process, 352
format command, 52
format of C programs, 204
format text, 137
formatted input/output, 201
formatting a disk, 352
formatting a text file, 89
formatting floppy disks, 156
formatting strings, 201
Fortran programming language, 352
fprintf function, 202, 204
FPU (floating point unit), 352
fragmentation, 125
fragments for storing files, 352
frame buffer memory, 26, 352
Frame Technology Corp., 352
FrameMaker desktop publishing program, 352
FrameViewer program, 352
free disk space, 137, 164
Free Software Foundation, Inc., 352
fsck command, 95, 108, 112, 113, 114, 175–176, 238, 239
fsflush daemon, 176
fstab file, 56, 238, 239
FTP (file transfer protocol), 241–244, 353
ftp command, 242
ftptool command, 242
full-duplex terminal session, 263
full-screen editor, 142
function key, 353
function overloading, 353
functions, 190, 353
functions in C, 199–200

G

GaAs (gallium arsenide) chip technology, 353
gain parameter, 172
gallium arsenide (GaAs) chip technology, 353
gamah2.par file, 270
games, 353
gateway machines, 227, 247, 353
gateway stations, 222
GB (gigabyte), 354
GCC_EXEC_PREFIX variable, 183, 207
gemcon host name, 228
Gemini tape format, 155
gets command, 211
getxvfonts command, 253
γH_2 (decoupler field strength), 270
GIF (graphical interchange format), 353
gigabyte (GB), 354
global parameter file, 168
global parameters, 116
global variables, 193

gloves for handling LHe, 320
GNU (GNU is Not UNIX) software, 354
GNU C compiler, 114, 190
GNU C compiler variables, 183
GNU C/C++ compilers, 354
GNU variable, 183
GNUDIR variable, 183, 206
go monitor command, 37, 94
gradient calibrations and performance tests, 299
graphical user interface (GUI), 100, 354
graphics controller, 25, 354
graphics terminals, 249
graphics variable, 183, 184
graphics workstation, 354
GraphOn terminals, 184, 249
grep command, 137
gripper abort, 166
gripper drops sample, 328
ground loop, 354
group file, 50, 245
groups file, 121
GUI (graphical user interface), 354
guru, 354
GX color graphics accelerator, 354

H

H1 file, 73
h1freq parameter, 72
h2cal command, 271
hacker, 355
HAL (Host-to-Acquisition Link) board, 28, 355
halt command, 96
handshaking, 90
handshaking lines, 355
hanging (stopped) system, 170
hangup status, 355
hard disk drives, 29, 355
    adding a drive, 35
    layout, 40
    partitions space defaults, 160
hard interrupt, 104
hard links to files, 127–128, 355
hardware, 355
hardware address, 225
hardware diagnostics, 93
hardware maintenance, 163
hcdept macro, 272
head command, 137
head crash, 355
header page, 88, 89
heat of vaporization, 318
helium contact with body, 18
helium flowmeter, 317
helium gas flowmeters caution, 20
helium gas storage cylinder, 323
help for operators, 355
help monitor command, 94
here documents, 211, 217
Hewlett-Packard Graphics Language (HP-GL), 356
hidden files, 51
hidecommand command, 75
hiding a command, 75
hierarchical file system, 355
hierarchy, 355
hierarchy of processes, 100
high-density floppy disks, 156
high-level programming language, 356
high-noise signal, 165
high-power amplifiers cautions, 20
High-Sierra File System (hsfs), 238
history command, 184
history length variable, 184
history of command lines, 356
history shell command, 138
history standard variable, 206
home directories, 49, 115, 117
home directory, 50, 71, 99, 111, 117, 134, 356
home file system, 107
HOME variable, 206
host address, 225
host address file, 257
host computer, 356
host computer name, 356
host computer number, 225, 356
host disk errors, 167
host name for acquisition CPU, 227
host name for Magnet and Sample Regulation board, 227
hostname command, 57, 138
hostname, changing the, 57, 170
hostname.* file, 228
hostname.ie0 file, 231
hostname.ie1 file, 247
hostname.le0 file, 231
hosts file, 62, 111, 170, 171, 225, 227, 231, 233, 245, 247, 264
hosts.equiv file, 228, 229, 231, 232
Host-to-Acquisition Link (HAL) board, 355
HotJava Web browser, 356
HP-GL (Hewlett-Packard Graphics Language), 356
HP-GL output, 86
HP-GL plotter output, 266
hsfs (High-Sierra File System), 238
HTML (HyperText Markup Language), 356
HTTP (HyperText Transfer Protocol), 356
hub, Ethernet, 357
HyperSPARC chip architecture, 357
HyperSPARC CPU chip, 24
hypertext link, 357
HyperText Markup Language (HTML), 356
HyperText Transfer Protocol (HTTP), 356
HZ variable, 206

I

I/O (input/output), 357
I/O ports, 29
ib_initdir directory, 115
IBM (International Business Machines, Inc.), 357
IBM PC computer, 249
IBM PCs, 245
IBM RS/6000, 249
    fonts required, 253
    using as X server, 253
ice blockage, 325
ice plugs, 321
icon, 357

id command, 138
ID of print request, 89
IDE (Integrated Device Electronics), 357
if statement, 197, 208, 214
ifdef statement, 192
ifndef statement, 192
ihwinfo command, 168
image browser setup files, 115
include directory, 114
include files, 191
include statement, 191
increasing space on floppies, 160
increment operators, 195
indent command, 185
indentation rules for C programs, 204
indirect detection experiments, 274
individual tape backups, 179
init command, 95, 96
init.d directory, 98
inittab. init file, 98
ink jet printer, 357
inline tap kit, 222
I-node (index-node), 357
I-node numbers, 126
I-node table, 109
inova host name, 227
inovaauto host name, 227
input from keyboard, 202
insert mode, 144, 357
insertion point, 358
install_cluster command, 48
installation of Sun workstations, 358
installboot command, 108
installing
    memory boards, 55
    new users, 49
    printers, 85
    second hard disk, 51
    Solaris, 39
    Solaris 2.5.1, 44
    Solaris 2.6, 48
    Solaris parts "after the fact", 58
    Solaris patches, 48
    Sun workstation, 35
    swap space, 56
int data type, 192
integer number, 192
integer value, 358
integration level, 358
interactive input, 211, 217
interactive mode for commands, 184
interface device, 358
interlacing the CRT display, 358
interleave factor (IL), 358
internal hard disk installation, 35
Internet address, 171, 225–227
Internet bibliography, 332–333
Internet network, 358
Internet Protocol (IP), 359
Internet Service Provider (ISP), 226, 359
internetworked network, 226
interpreter program, 358
interrupt signal, 359
intranet network, 359
invisible file, 359
ionic samples, 270
IP (Internet Protocol), 359
IP address of router, 233
IRIX operating system, 359
ISDN (Integrated Systems Data Network), 359
ISDN port, 27
ISP (Internet Service Provider), 359

J

Java programming language, 359
JavaScript, 360
jexp command, 116
jobs shell command, 139
Joy, William, 360
JPEG (Joint Photographic Experts Group), 360
jukebox device, 360
jumper, 360

K

KB (kilobyte), 360
Kermit protocol, 266
kernel directory, 112
kernel program block, 360
keyboard, 360
keyboard input, 132, 202
keyboard layouts, 114
keyboard shortcuts, 134
kill command, 66, 68, 138
killacqproc command, 65
killing a program, 360
kilobyte (KB), 360
Korn shell, 50

L

L1 key, 69, 170
label command, 53
LAN (local-area network), 222, 360
laser printer, 360
LaserJet printer, 86, 360
last line mode of vi, 144
latency time, 360
latent heat of vaporization, 319
LD_LIBRARY_PATH variable, 102, 186, 206
left-shift all arguments, 210
level 0, 93–94
level 0 and level 1 dumps, 176
level 1 run mode, 95
level 2 run mode, 95
level 3 and level 5 dumps, 178
level 3 run mode, 96
Lexmark color printer, 360
lf command, 51, 135
lf file prefix for mail, 265
lib directory, 112, 114
library files, 190
license manager, 361
limNET Ethernet protocol, 245, 361
line conditioner, 361
linear shim Z1, 276

Index

line-oriented text editors, 147
lineshape check, 317
link to a file, 136
linked network, 226
linker program, 361
linking, 203
linking to VNMR software directory, 115
lint command, 189, 192
liquid helium
    dewar service, 323
    handling, 318
    handling safety, 320
    level in dewar, 317
    measuring, 320
    safety precautions, 320
    storage containers, 320
    transferring, 321
list files, 135
ll command, 135
ln command, 55, 127, 136
local area network (LAN), 360
local host, 225
local maxima, 283
local variables, 193
local-area network (LAN), 222
localhost host name, 227
lock type, 116
lock_1.primary file, 116
lockscreen command, 163
log book, 272
log file, 113
log out automatically, 184
log out of system, 138
loghost host name, 227
logical operators, 196
login process, 99, 122, 361
login shell, 361
login shell script, 183
LOGNAME variable, 206
logout command, 70, 184, 263
logout process, 361
long data type, 192
looping, 209, 215
looping flow control, 198–199
lost+found directory, 112, 113, 114, 160, 176
low-density floppy disks, 156
low-noise signal, 165
low-order shims, 281
lp command, 87, 89
lp device, 86
lp directory, 111, 113
lp script, 98
LPDEST variable, 89
lpr command, 88
ls command, 51, 71, 119, 132, 133, 135, 158, 183, 184
lunchbox storage pack, 361

M

Macintosh as X server, 254
Macintosh computer, 249
maclib directory, 117, 128
maclibpath parameter, 117
macro file, 361
macroedit command, 141
macrovi command, 141
MAGICAL (MAGnetics Instrument Control and Analysis Language), 361
MAGICAL command language, 103
MAGICAL macros for spectral editing, 272
Magnet and Sample Regulation (MSR) board, 112
    host name, 227
magnet quench cause, 321
magnet quench warning, 18
magnetic media caution, 19
magnetic tape, 150
magneto-optical disk (MOD), 362
mail administration files, 111
mail client machine, 264
mail command, 185
mail directory, 111, 113
mail problems, 265
mail software configuration, 185
mail tool, 362
Mail Tool program, 185
MAIL variable, 206
mailhost, 264
mailtool command, 185, 264
main function, 190
maintenance contract, 362
maintenance of Sun workstations, 362
makesuacqproc command, 67, 123
makeuser command, 50, 71, 122, 183
man command, 132
man directory, 114
MANPATH variable, 183
manual directory, 128, 165
manual notational conventions, 22
Manual Shim menu, 74
manuals, importance to users, 362
mass storage device, 362
Master NIS file server system, 171
master server, 245
math.h include library, 203
mathematical operators, 195
MAXSTR constant, 191
MB (megabyte), 363
MBus (Memory Bus), 362
MBus system interconnect, 26
MC68000 CPU, 362
MC68020 CPU, 362
MC68030 CPU, 362
MC68040 CPU, 363
MC68881, MC68882 coprocessors, 363
mean time between failures (MTBF), 366
measuring liquid helium in a container, 321
megabyte (MB), 363
megaflops (million floating point operations per second), 363
melting point of helium and nitrogen, 318
memory boards installation, 55
memory buffer exceeded, 170
memory management unit (MMU), 364
memsize variable, 40, 56, 183, 206
menthol sample, 271
menu, 363
menu button, 363

message of the day file, 51
metal objects warning, 17
mezzanine board, 363
MFlops (million floating point operations per second), 363
microprocessor, 363
MicroSPARC CPU chip, 24, 363
MII (Media Independent Interface), 364
million instructions per second (MIPS), 364
mini UNIX, 364
minicomputer, 364
minus sign notation, 131
MIPS (million instructions per second), 364
MIPS CPU chips, 364
mkdir command, 135
mkfile command, 56, 169
MMU (memory management unit), 364
MOD drive, 364
modem (modulator-demodulator), 364
modem connections, 261
modem types, 263
modifying the instrument, 18
modulation frequency parameter, 270
molecular weight of helium and nitrogen, 318
monitor diagnostic, 364
monitor mode, 93–95
monitor screen, 364
monthly maintenance, 317
more command, 137, 184
Mosaic Web browser, 365
motd file, 51
motherboard, 365
Motif graphical user interface, 365
Motif interface, 101
Motif window system, 249
Motorola 68000 CPU family, 29
Motorola CPU chips, 365
mount command, 55, 57, 158, 159, 234, 238
mount directory, 234, 365
mount point creation, 55
mountall command, 239, 246
mountd daemon, 237
mounting, 365
mounting file systems, 234
mounting floppy disks, 158, 235
mounting limitations, 237
mounting removable media, 123
mouse, 365
mouse buttons, 365
mousepad, 365
move files, 136
movesw macro, 269
movetof macro, 269
MPEG (Motion Picture Experts Group), 365
mqueue file, 264, 265
MS-DOS (Microsoft Disk Operating System), 366
MS-DOS floppy disk format, 157
MTBF (mean time between failures), 366
mult parameter, 272
multimedia applications, 366
multiple hard links to a file, 127
multiprocessor system, 366
multisegment tapes, 154
multitasking system, 366
multithreading, 366
multiuser backups, 179
multiuser mode, 93
multiuser system, 366
mv command, 136, 237
mwm window manager, 255

N

n monitor command, 94
name servers, 233
name variables, 212
nd.local file, 171
net directory, 112, 240
netmask, 226
netmasks file, 227
Netra workstation, 366
Netscape Navigator Web browser, 366
netstat command, 172
network, 366
Network File System (NFS), 235, 367
Network Information Service (NIS), 245, 367
Network Information Services (NIS), 171
network number, 225, 226
network security bibliography, 332
network system security, 228
networking, 221–247
networking bibliography, 331–332
networks file, 245
new user assignment, 49
newfs command, 109, 159, 160, 176
NFS (Network File System), 245, 367
NFS client machine, 236
nfs directory, 237
NFS mounting, 111, 246
NFS mounting options, 236–237
nfsd daemons, 237
NFS-type mounting, 235
NIS (Network Information Service), 367
NIS clients, 245
NIS service, 57
nitrogen contact with body, 18
nitrogen flowmeter, 317
nitrogen gas flowmeters caution, 20
nmr group, 122
NMR system administrator, 117
node in network, 367
nodename file, 228
non-spin shims, 279
nonvolatile RAM (NVRAM), 27
notational conventions, 22
notebooks for system, 318
notrouter file, 247
np parameter, 167, 172
nroff command, 132, 137
nsswitch.conf file, 233
nt parameter, 172
N-type connectors, 222
nucleus standard parameter sets, 73
nucleus table, 72
nuctables directory, 72
null character, 194
null device, 367
null modem, 89–90, 262, 263, 367

null modem cable, 367
null string, 194
number-sign notation, 205, 212
NVRAM (nonvolatile RAM), 27, 367

O

object file, 367
object libraries, 114
Object-Oriented Programming Style (OOPS), 368
objects, 368
odd-order shims, 277
off-line task, 368
off-resonance decoupling, 270
ok prompt, 94
old-mode monitor command, 94
OLIT (OpenLook Intrinsics Toolkit), 368
olwm command, 102
olwm window manager, 255
online process, 368
OOPS (Object-Oriented Programming Style), 368
open access files, 368
Open Look, 368
  configuration file, 186
  file manager, 187
  root menu definition, 187
  screen background, 186
  shortcuts, 139
  textedit text editor, 141
  window manager, 102, 186
  window system, 249
Open Systems Foundation (OSF), 369
OpenGL application programming interface, 368
OpenLook Intrinsics Toolkit (OLIT), 368
openwin directory, 102, 114
openwin shell script, 102
OpenWindows, 100, 368
  configuration file, 186
  deskset utilities, 101
  environment, 102, 183
  fonts possible, 188
  initialization, 186
  software directory, 114
  textedit program, 141
OPENWINHOME variable, 102, 186, 206
operating system program, 368
Operations Log notebook, 318
operator interface devices, 369
operator overloading, 369
opt file system, 107
opt slice, 113
optical media mounting, 123
optimization level for C compiler, 203
optional software slice, 113
organic solvents, 270
OSF (Open Systems Foundation), 369
output to a file, 202
output to screen or window, 201
owner of file, 369

P

p1 parameter, 172
pacemaker warning, 17
packet collision information, 172
packets, 230, 369
page command, 137
paging, 369
parallel computing, 369
parallel I/O, 369
parallel I/O controller, 27
parallel port, 86
parameter sets for a nucleus, 73
parent directory, 134, 369
parentheses notation, 132
parity checking, 369
partition, 369
partition command, 53
partition map, 108
partitioning the disk, 168
partitions on hard disks, 105
Pascal programming language, 369
Pascal vs. C programming languages, 189
passwd command, 48
passwd file, 50, 99, 111, 117, 121, 245
password, 370
password access, 228
password aging mechanism, 169
password assignment, 48
password checking, 99
password file, 117
password invalid, 169
password protection, 164
password requirements, 49
password selection, 230
patches installation log, 113
path for shell scripts, 218
path to a file, 133
PATH variable, 145, 206
path variable, 102, 183, 186
PC compatibles, 245
PC computers as X server, 254
pcfs directory, 158
PCI bus, 26, 370
PCI bus controller, 24
PCI ports, 27
PCL output, 86
PCMCIA (Personal Computer Memory Card Interface Association), 370
PDF (Portable Document Format), 370
peak asymmetry, 277
peak broad base, 279
peak broadening, 277
performance measures, 370
performance meter, 101, 224
performance, computers, 370
peripheral, 370
peripheral component interconnect. See PCI
Perl programming language, 371
permission flags, 120
permission to access file, 371
personal computers
  starting VNMR, 254
  X server software, 254
PGX graphics accelerator, 371
phase stability, 305
piggyback board, 371
ping command, 172, 224, 230

pipe, 133, 371
pixel, 371
pizza box CPU module, 371
pkgadd command, 58
platform directory, 112
plotter definition, 85
plotter device, 371
plotter parameter, 86
plotters characteristics, 86
PlotView software package, 266
pointer variable, 193
pointers in C, 200–201
Point-to-Point Protocol (PPP), 372
Point-to-Point Thinnet Network, 222
polar solvents, 270
polarization transfer, 271
port, 371
portability, 371
PostScript output, 86
PostScript page description language, 371
power failure, 175
power failure effects, 371
power switch, 372
power switch turnoff order, 69
pp parameter, 271, 272, 273
ppcal macro, 273
PPCAL pulse sequence, 273
ppmm parameter, 86
PPP (Point-to-Point Protocol), 372
precision of data word, 372
precooling the transfer tube, 321, 323
preprocessor, 191
preprocessor program, 190, 203
pressure regulator, 323
pressure-relief valves, 320
preventative maintenance, 317
primary cache memory, 24
primary group membership, 121
print spooling, 42
Printcap entry, 86
printcap file, 372
printenv monitor command, 94
printer administration files, 111
printer definition, 85
printer device, 372
printer installation, 85
printer ports, 91
PRINTER variable, 183
printers characteristics, 86
printf function, 201
printing and plotting under VNMR, 87–88
printon, printoff commands, 88
probe-scsi monitor command, 37
proc directory, 112
Proc family, 61, 65
procedures definition, 218
Procedures Log notebook, 318
process families, 100
process in program execution, 372
processes list, 112
processing power, 29
procpar file, 180
PROM-based software level, 93
prompt, 372
prompt format, 184
prompt standard variable, 206
props button, 149
prosthetic parts warning, 17
protection bits, 119, 119–121
protocol, 372
proton decoupler pulse calibration using PPCAL, 273
proton decoupler pulse width, 273
prtvtoc command, 108, 138
ps command, 66, 138
psg directory, 191
psggen command, 190
public data networks, 261
public e-mail access, 227
pulse sequence program structure, 191
pulse sequence programming, 189, 190, 203
pulse sequence variable definitions, 194
pulse width calibration, 269
pulsesequence function, 191
pw parameter, 167, 172
pw90 value, 269
pwd command, 133, 135
PWD variable, 206
pwx1 parameter, 274
pwxcal macro, 274
PWXCAL pulse sequence, 274

Q

qf file prefix for mail, 265
QIC (quarter-inch cassette), 372
QIC tape drives, 33
QIC tape format, 151
Qtune file, 258
quench, magnet, 321, 323
quitting the shell window, 103

R

radial shims, 283
radio-frequency emission regulations, 20
RAID (redundant arrays of inexpensive disks), 372
RAM (random-access memory), 25, 373
raster entry, 86
raw devices, 105, 235
rc files, 183
rc.boot file, 171
rc0 to rcS startup scripts, 97
rcp command, 228, 229, 231, 232, 241, 245
rdsk directory, 105
read files to tape, 149
read permission, 119
read tape, 136
read/write optical disk drives, 35
reading from a tar file, 152
reading tapes of any format, 155
real-time clock (TOD), 27
real-time variables, 193
reboot command, 55, 96
rebooting the acquisition system, 67
rebooting UNIX, 69
receiver overflow warning, 165
receiving mail, 264

recovering damaged files, 178
recursion, 373
recursive copy of a file, 136
reexecute last command line, 138
refresh rate, 373
register variables, 194
rehash command, 218
relational operators, 196
relative paths, 133
relaxation time, 269
release of software, 373
relief valves warning, 19
remote computer actions, 374
remote control files, 183
remote file, 91, 263
remote file system mounting, 235
remote home directories, 246
remote host commands, 232
remote login, 170, 229
remote login on a remote host, 232
remote printing and plotting, 232
remote tape drive, 154—155
remote X server, 184
remote X terminals, 186
removable magnetic tape, 150
removable media mounting, 123
removable quench tubes warning, 19
remove
    alias, 138
    directory, 135
    files, 136
reset monitor command, 94
resetting the console, 170
residual couplings in dioxane, 271
resizable windows, 71
resolv.conf file, 233
restarting acquisition processes, 65
restarting UNIX, 69
restore command, 155
Return key, 22, 131
REXEC command protocol, 255
rf field homogeneity of decoupler, 274
RF homogeneity tests, 299
rftype parameter, 72
RG/58U coaxial cable, 223
rgb.txt file, 188
RISC (reduced instruction set computer), 374
RJ45 connector, 374
rlogin command, 170, 172, 228, 231, 232, 245, 252, 253
rm command, 127, 136, 184
rmdir command, 135
ROM (read-only memory), 374
root access to NFS mounted systems, 237
root login name, 48
root menu, 374
root partition, 107
root partition size, 265
root password, 48, 67, 95
root slice, 41, 110
root superuser, 374
rooted environment, 254
rootmenu, 102
rotational latency, 108
route command, 233
routine shims, 281
RS/6000 computer. See IBM RS/6000
RS-232 cable, 166
RS-232 expansion board, 89
RS-232 ports, 249
RS-232 system connection, 261
RS-232, RS-423 serial communications, 374
rsh command, 228, 231, 232, 245, 253
RSH command protocal, 255
run levels, 93
run-time library, 114
run-time math library, 203

S

sadm directory, 113
safety concerns, 374
safety precautions, 17, 19, 320
safety vent valve, 322
sample changer, 328
sample changer errors, 166
sample scoring problems, 327
Save Workspace option, 187
sbin directory, 112, 114
SBus internal bus, 375
SBus ports, 27
scanf function, 202
scaning the file system, 175
SCCSid static variable, 203
scheduler program, 375
screen blanking, 375
screen burn-in, 163, 375
screen locking, 375
scroll bar, 375
SCSI (Small Computer Systems Interface), 375
SCSI address verification, 36
SCSI addresses, 28
SCSI bus, 27, 106
SCSI communications troubleshooting, 170
SCSI communications unreliable, 170
SCSI controller, 26
SCSI disks, 105
SCSI driver box, 28
SCSI errors, 167
SCSI ID selection, 35
SCSI-3 connectors, 28
SCSI-ID, 106
search for pattern in file, 137
secondary cache memory, 24
secondary lock file, 116
sector of disk, 375
security patches, 123
sed command, 87, 137
seek time, 375
semicolon command separator, 131
seqgen command, 190
seqgen shell script, 203
serial I/O, 375
serial I/O controller, 27
serial link between systems, 261
serial port, 86
server computer, 376
set command, 134

set statement, 206
setacq command, 66, 167
setenv command, 94, 207, 253
set-GID bit, 122
setnoether command, 224
set-UID bit, 122
setup macro, 73
SGI (Silicon Graphics, Inc.), 376
SGML (Standard Generalized Markup Language), 376
shadow file, 50, 99, 245
shaped pulse test descriptions (channels 1 and 2), 304
share command, 55, 236, 238
share directory, 114
shareall command, 55, 237
sharetab file, 55
shell, 376
shell command interface, 50
shell for commands, 132
shell script, 376
shell scripts, 205
shell tools, 102, 103
SHELL variable, 206
shelltool command, 102
shielded cable, 376
shielded twisted pair (STP), 381
shielding for rf, 376
shift command, 210
SHIM and FID display, 74
shim buttons selected by users, 73
shimming errors, 167
shimming procedures, 275
shims
  asymmetry, 277
  broadening of the peak, 277
  complex lineshapes, 278
  effects on symmetry of the resonance, 275
  even-order, 277
  first-order spinning sidebands, 279
  illustration of shim interactions, 275
  linear, 276
  non-spin, 279
  odd-order, 277
  removing spinning sidebands, 282
  second-order spinning sidebands, 279
  setting high-order axial shims, 282
  setting high-order radial shims, 283
  setting low-order or routine, 281
  shapes of the axial gradients, 275
  theoretically predicted changes in lineshape, 275
  typical interactions for axial shims, 281
  what are they, 275
  when more than one are misadjusted, 278
shims file, 49
shimset parameter, 73
short data type, 192
showconsole command, 172
showrev command, 48, 123
shutdown command, 51, 96, 104
shutting down
  acquisition computer or cabinet, 104
  entire system, 104
  multiuser system, 51
  UNIX, 68
signal-to-noise check, 317
Silicon Graphics (SGI) workstation, 249
Silicon Graphics, Inc. (SGI), 376
SIMM (single in-line memory module), 376
single-slice installation, 42, 44
single-user mode, 93, 95, 376
skin damage from helium and nitrogen, 319
slash notation, 133
slave servers, 245
slice, 377
slice number, 106
slices of hard disks, 105
SLIP (Serial Line Internet Protocol), 377
SMCC (Sun Microsystems Computer Corp.), 377
SMD (surface mounted device), 377
SMS sample changer, 328
software handshaking, 90
software installation log, 113
software maintenance, 163, 377
software types, 377
SoftWindows 95, 377
Solaris 1.1 operating system, 377
Solaris 2.5.1 disk layout, 40–43
Solaris 2.5.1 installation, 44–48
Solaris 2.6 disk layout, 43–44
Solaris 2.x options, 40
Solaris 2.x software, 28
Solaris 2.x software environment, 377
Solaris CD-ROM, 44
Solaris installation "after the fact", 58
Solaris patch installation, 48
Solaris patches installation log, 113
solids high-power amplifiers caution, 20
Solstice network software, 378
sort command, 137
sort lines in text file, 137
Source Code Control System, 203, 204
source command, 186
source files, 114
SPARC (scalable processor architecture), 378
SPARC microprocessors, 29
SPARCclassic workstation, 378
SPARCstation 1 and 1+ workstations, 378
SPARCstation 1+ or 1 power on, 70
SPARCstation 10 workstations, 378
SPARCstation 2 workstation, 379
SPARCstation 20 workstations, 379
SPARCstation 330 workstation, 379
SPARCstation 4 workstations, 379
SPARCstation 5 workstations, 379
SPARCstation ELC workstation, 254, 380
SPARCstation IPC workstation, 380
SPARCstation IPX workstation, 380
SPARCstation LX workstation, 380
SPARCstation SLC workstation, 380
SPARCstation workstations, 378
SPEC (System Performance Evaluation Cooperative), 380
SPECfp92 performance measure, 380
SPECfp95 performance measure, 381
SPECint92 performance measure, 381
SPECint95 performance measure, 381

SPECmark performance measure, 381
SPECrate performance measure, 381
spectral editing, 272
spectrometer host, 247
spin test, 327
spinner does not rotate, 328
spinner errors, 165
spinning instabilities, 327
spinning sidebands, 279
spinning sidebands, removing, 282
spool directory, 113
spooled data storage, 113
spooling directory, 265
spray command, 230
square brackets wildcard notation, 134
SRAM (static RAM), 381
stack register, 381
staff group, 122
stand-alone workstation, 381
standard environment for VNMR, 71
standard input, 132
standard output device, 201
standard parameter sets for a nucleus, 73
standard printer, 86
standard.h file, 191
STAPE command, 155
startacqproc command, 66, 70, 104
starting
  Acqproc process, 66
  VNMR, 103
  VnmrI, 250
startup commands, 112
statement separator, 197
static discharge, 381
static variables, 194
status field parameter strings, 191
stdio.h file, 190, 191
stdpar directory, 73
steady-state, 269
sticky bit, 122
stinger, 323, 326
Stop key, 69, 170
Stop-a key combination, 94, 104
stopping Acqproc process, 65
stopping acquisition processes, 65
storage class for variables, 193
storage dewar, 322
storing directories on tape, 149
STP (shielded twisted pair), 381
streaming tape mode, 151
string variables, 194
strings as a data type, 193
struct data type, 193
structured programming, 381
structures, 193
stty command, 251, 262
Style Manager, 101
su acqproc command, 67, 104, 173
su command, 137
subnet references, 167
suffix to filename, 133
summary block, 109
sum-to-memory error, 167
Sun as X server to VnmrSGI, 254
Sun ELC as X server, 254
Sun FPA (floating point accelerator), 127
Sun Microsystems, 383
Sun Microsystems Computer Corp. (SMCC), 377
Sun patch FTP server, 48
Sun terminal, 184
Sun workstation, 23–28
Sun workstation documentation, 329
Sun workstation installation, 35
Sun-2 workstations, 382
Sun-3 workstations, 28, 382
Sun-4 workstations, 382
suninstall command, 225, 227
SunOS 4.1.x, 267
SunOS 4.1.x to Solaris 2.x transition, 267
SunOS 4.x operating system, 383
SunPC, 383
SunSoft C compiler, 190
SunSoft division, 383
SunSolve Web site, 48
Suntools, 383
SunView graphical user interface, 383
SunWindows graphical user interface, 383
superblock, 108
Supercache memory, 383
supermini computers, 383
SuperSPARC CPU chips, 24, 383
superuser, 383
surface analysis, 53, 383
surface mounted device (SMD), 377
sw parameter, 172
swap command, 56, 113, 168
swap space, 40, 44, 56, 107, 112, 168, 384
swapping processes, 384
swept fm decoupling, 270
switch statement, 198
SX graphics accelerator, 26, 384
symbolic links, 55, 128–130, 136, 152, 384
sync command, 69, 94, 138, 175
synchronize system disk, 94, 138
synchronous data transfer, 262
synchronous transmission of data, 384
syntax checking, 189
syslog file, 113
system administration commands, 114, 137
system administration files, 113
system interconnect, 26
System Log notebook, 318
system messages, 51
system processes list, 112
system recovery commands, 112
system shutdown, 104
system startup scripts, 97
System V UNIX, 106, 384
sys-unconfig command, 57, 170, 225

T

T junction, 222
T_1 time, 269
tail command, 137
tap kit, 222, 385
tape backup, 164, 175, 176
tape backups, 317

tape cassette, 385
tape catalogs, 152
tape command, 155
tape device configuration, 150
tape drive on remote system, 154
tape drives, 33, 385
tape lengths, 177
tape magnetic medium, 385
tape rewinding, 154
tape storage, 149
Tape Tool program, 149
TAPE variable, 150, 153, 183
tar command, 55, 130, 131, 136, 149, 157, 164, 179, 180
tar command (UNIX), 317
tar file, 150, 151
tar file system, 157
tar tape archive program, 385
target address or ID, 105
task, 385
TB (terabyte), 386
Tcl/Tk (Tool command language /Tool kit), 385
Tcl/Tk scripting language, 219
Tcl/Tk variables, 183
TCL_LIBRARY variable, 183
TCP (transport control protocol), 385
TCP/IP (transport control protocol/internet protocol), 245, 386
Tektronics terminals, 184
Tektronix terminals, 249
telnet, 254
  client startup, 256
  session startup, 255–256
temporary files, 112
terabyte (TB), 386
TERM signal, 138
TERM variable, 146, 206
term variable, 102, 146, 184, 206
termcap file, 146
terminal emulation, 249
terminal emulation software, 266
terminal information database, 114
terminal interface, 386
terminal types, 184
terminate processes, 138
terminator, 386
terminator device, 27, 223
test function, 207, 213, 214
text commands, 137
text editing programs, 141–147
text file, 386
text file compression, 180
text processing, 386
textedit command, 137, 139, 141
textedit text editor, 386
tftpboot directory, 112
TGX (TurboGX) color graphics accelerator, 386
then statement, 209
thermal acoustical liquid level indicator, 320
thick Ethernet cabling, 222, 223
thin Ethernet, 386
thin Ethernet coaxial cable, 223
Thinnet Ethernet cable standard, 222
Thinnet network, 223
TIFF, 386
tilde sign notation, 134, 263
time sharing operating system, 386
time zone definitions, 114
tip command, 263
TK_LIBRARY variable, 183
tmp directory, 87, 113
tmp file system, 112
tmp partition fills up, 169
tn parameter, 72
tof parameter, 172
token replacement, 191
tool, 386
toolbar, 101
top-down programming, 386
touch command, 104, 136
tpwr parameter, 172
tr command, 137
transceiver, 222, 387
transceiver for Ethernet, 223
transfer line, 321
transfer rate for Ethernet, 231
transfer rate over interface, 387
transfer tube, 320, 322, 323
transferring directories, 150
transferring large files via floppies, 160
transferring LHe from a storage container, 321
transition to Solaris 2.x, 267
translate characters, 137
transmitter frequency from nucleus tables, 72
Trash directory, 187
tree structure for a file system, 124
tree structure of directories and files, 387
triple-resonance experiments, 274
troff command, 137
troubleshooting, 165
  acquisition status codes, 165
  Ethernet, 171
  sample changer, 328
  system hangs, 170
troubleshooting activities, 387
trusted hosts, 228
T-switch, 384
ttya file, 262
ttya, ttyb ports, 86
ttymon process, 99
tunefs command, 109
TurboSPARC CPU chip, 24, 387
twisted pair (TP) port, 26
twisted-pair Ethernet, 387
twisted-pair Ethernet (TPE), 221, 224
two dots file name notation, 134
type casting operator, 196
type command, 52
TZ variable, 206

U

ucb directory, 114
ucbinclude directory, 114
ucblib directory, 114
UDP (user datagram protocol), 387
ufs (UNIX file system), 106, 108
  parameters, 109

ufsdump command, 149, 176, 177
ufsrestore command, 149, 178, 179
Ultra 1 workstations, 388
Ultra 10 workstation, 388
Ultra 2 workstations, 388
Ultra 30 workstations, 388
Ultra 5 workstation, 388
Ultra 60 workstation, 389
Ultra IIi CPU chip, 26
Ultra•nmr Shims, 275
UltraSCSI controller, 26
UltraSPARC CPU chip, 24, 389
UltraSPARC Port Architecture (UPA), 26, 390
umask file protection system, 74
umount command, 158, 235, 236, 239
umountall command, 57, 239
Unaccounted Users list, 81
unalias command, 138
uname command, 138
uncompress command, 138, 180, 181
uncompress file, 138
unconditional looping, 209, 215
undef statement, 192
unformat floppy using stray field, 157
UniPack enclosure, 389
Unipack modules, 27
UNIX, 389
    administration files, 111
    argument separator, 131
    bibliography, 329–330
    command names, 131
    command separator, 131
    documentation, 329
    emergency shutdown, 69
    file name rules, 133
    file system, 106, 158
    file system creation, 55
    file system generation, 159
    file system tree, 124
    global environment variables, 183
    important commands, 135
    kernel, 112
    manuals, 114
    online manuals, 329
    programming bibliography, 333–334
    restarting, 69
    security bibliography, 332
    shell interface, 102
    software file system, 114
    system administrator, 48, 50, 51
    tools bibliography, 330–331
UNIX International (UI), 389
unix2dos command, 158
UNIX-to-UNIX copy, 263
unknown host message, 257
unlock command, 116
unset command, 134
unshare command, 237
unshielded twisted pair (UTP), 391
unsigned int data type, 192
unsigned integer number, 192
unsigned short data type, 192
UPA (UltraSPARC Port Architecture), 390
update daemon, 176
updates file dates, 136
upfield tail on the peak, 277
upgrading Sun workstations, 390
upper barrel warning, 18
URL (uniform resource locator), 390
usable disk space, 109
user accounting log, 76
user file, 179
user of UNIX software, 390
user partition, 177, 391
user password file, 50
user standard parameters, 73
user standard variable, 206
USER variable, 206
user_templates directory, 183
users currently logged in, 138
usr file system, 107, 114
utility programs, 391
UTP (unshielded twisted pair), 391
uucp command, 263, 264

V

V-24 connector, 27
vampire tap kit, 222
vapor density of helium, 319
vaporization property of helium, 319
var file system, 107
var slice, 113
variable declarations, 193
variable initialization, 193
varian.xicon file, 186
VAX computer under VMS, 245
venting helium vapor, 319
vertical bar notation, 133
vfstab file, 55, 158, 160, 235, 238, 246
vi command, 137
vi text editor, 102, 142, 185, 391
    command mode, 142
    insert mode, 143, 144
    last line mode, 144
    problems, 145
    running remotely, 146
    shell calls, 146
video technology tape drives, 33
virtual file system, 108
virtual memory, 391
virtual memory management, 24
virtual terminals, 250, 251, 253
virus software bug, 391
VIS (Visual Instruction Set), 391
VME bus structure, 391
vmunix file, 112
vn command, 184
VNMR, 391
    accounting window, 84
    application configuration files, 187
    argument separator, 131
    customization files, 183
    file name extensions, 133
    multiuser setup, 117
    patch installation, 48
    programs, 103
    resizable windows, 71

- shell, 103
- software directory, 115
- source code kit, 189, 190
- startup from personal computer, 254
- system variables, 183
- user’s environment, 71
- user’s home directories, 115
- workspaces, 116

vnmr command, 250, 251, 252, 253
vnmr directory, 115, 117
VNMR Online, 101
VNMR User Programming Manual, 189
vnmr_accounting command, 77
vnmr1 file ownership, 121
vnmr1 NMR system administrator, 138
vnmr1 user name, 117
vnmreditor variable, 141, 183, 206
VnmrI, 249, 250
- exiting, 251
- running multiple copies, 251
- startup, 250

vnmri command, 250
VnmrI software package, 391
vnmrmenu variable, 183, 206
vnmrplot shell script, 88
vnmrprint shell script, 86, 87–88
VnmrS software package, 391
VnmrSGI, 249, 250, 392
- exiting, 252
- moving windows, 252
- resizing windows, 252
- startup, 251

vnmrsgi command, 251
vnmrsys directory, 115, 117
vnmrsystem variable, 115, 183, 206
vnmrtext variable, 183
vnmruser variable, 183, 206
VnmrV software package, 392
vnmrwish directory, 219
VnmrX, 249
VnmrX software package, 392
void type, 192
vol directory, 112
vold process, 157
vold volume daemon, 123
volmgt file, 161
volmgt script, 98
volume management, 98, 161
volume manager, 112
VRAM memory chips, 392
VRML (Virtual Reality Modeling Language), 392
VT errors, 166
VT experiment warning, 18
vttime parameter, 166
vxBoot file, 62
VXR-4000 tape format, 155
VxWorks operating system, 61, 112

W

WABI (Windows Applications Binary Interface), 392
Wabi (Windows Applications Binary Interface), 41
WALTZ-16 decoupling, 270
WAN (wide-area network), 393
warning error codes, 165
warnings defined, 17
wastebasket, 187
wbs command, 165
wc command, 132
Web (WWW, World Wide Web, W3), 394
Web authoring, 393
Web authoring tool, 393
Web browser, 393
Web browsers and color flashing, 101
Web page, 393
Web server, 393
weekly maintenance, 317
weight of nitrogen-shielded container, 319
weighting functions, 190
werr command, 165
wexp command, 165
while loop, 198
while statement, 209, 215
who am i command, 138
who command, 138
whois command, 50
wide-area networking (WAN), 266
width of the field for printing, 202
wildcard characters, 134
Winchester disk, 393
windowless multiuser mode, 100
windows, 393
windows resizing, 71
wnt command, 165
word count, 132
word data size, 393
word processing, 394
working directory, 133
workstation, 394
World Wide Web (W3, Web, WWW), 394
WORM drive, 394
WORM drives, 35
wrap text lines, 137
write files, 136
write files to tape, 149
write permission, 119
wtgen command, 190
WWW (World Wide Web, W3, Web), 394
WYSIWYG (what you see is what you get), 394

X

X client, 249, 394
X decoupler pulse calibration using PSXCAL, 274
X host, 249
X server, 184, 249–250, 394
- setting up, 252

X terminal, 186, 250, 395
X Toolkit (Xt), 395
X Window System, 100, 249, 395
- bibliography, 334–335
- customizing the X environment, 258

X11 defaults, 102
X2–Y2 non-spin shim, 279
xcharp1 parameter, 86
xdm graphical login, 96, 183
xf file prefix for mail, 265

XFILESEARCHPATH variable, 206
xfn directory, 112
xhost command, 184, 252
xinit command, 102
xlsfonts command, 188
XON/XOFF data control, 394
x-ray emission from CRT, 394
Xt (X Toolkit), 395
xterm window, 255–257
    controlling appearance, 257
    displaying, 256
xvfonts directory, 253, 254
XY non-spin shim, 279

Y

Y non-spin shim, 279
ycharp1 parameter, 86
year 2000 compliance, 395
yellow pages service, 57, 245
ypwhich command, 171

Z

Z1 linear shim, 276
Z2 even-order shim, 277
Z3 odd-order shim, 277
Z4 even-order shim, 277
Z5 odd-order shim, 277
zcat command, 182
zero latency drive, 108, 395
zero wait-state, 395
ZX non-spin shim, 279
ZY non-spin shim, 279