ULTRIX/UWS

Release Notes

Order Number: AA-ME85D-TE

Product Version:	ULTRIX/UWS Version 4.1
Operating System and Version:	ULTRIX Version 4.1

This manual contains notes critical to the installation and use of ULTRIX/UWS Version 4.1. Read these notes before installing ULTRIX/UWS Version 4.1.

digital equipment corporation
maynard, massachusetts

Contents

About This Manual

1 Installation Notes

2 Processor-Specific Notes

3 ULTRIX Software Notes

4 ULTRIX Worksystem Software Notes

5 Layered Products Notes

6 Documentation Notes

A Problems Resolved Since Last Release

B Changes and New Features in Version 4.1

C Changes and New Features in Version 4.0

D RISC-VAX System Differences and Porting Hints

Figures

Tables

About This Manual

These release notes list the major new features and changes to the software and documentation for ULTRIX/UWS Version 4.1. The release notes also describe problems in the ULTRIX/UWS Version 4.1 software and documentation that were discovered too late to document elsewhere. The changes and new features described here are based on how ULTRIX/UWS Version 4.1 differs from ULTRIX/UWS Version 4.0 and how ULTRIX/UWS Version 4.0 differs from ULTRIX–32 Version 3.1 and UWS Versions 2.1 and 2.2.

Read these release notes before you install the ULTRIX/UWS Version 4.1 software.

If you discover errors, omissions, or inaccuracies as you use the software and documentation, submit a Software Performance Report (SPR).

Audience

This document is written for the person responsible for installing, managing, and maintaining the ULTRIX/UWS Version 4.1 operating system and its documentation. Programmers and other users of ULTRIX/UWS Version 4.1 facilities will find information in these release notes that affects their work as well.

Organization

This document contains six chapters and four appendixes:

Chapter 1 – Installation Notes
: Discusses the overall installation of ULTRIX/UWS Version 4.1, and provides workarounds to software and hardware problems, if workarounds exist.

Chapter 2 – Processor–Specific Notes
: Provides installation instructions and notes, as well as workarounds for software and hardware problems for specific processors.

Chapter 3 – ULTRIX Software Notes
: Discusses problems with the ULTRIX operating system and provides workarounds to these problems, if workarounds exist.

Chapter 4 – ULTRIX Worksystem Software Notes
: Discusses problems with ULTRIX Worksystem Software and provides workarounds to these problems, if workarounds exist.

Chapter 5 – Layered Products Notes
: Discusses problems with layered products and provides workarounds to these problems, if workarounds exist.

Chapter 6 – Documentation Notes
: Discusses problems with ULTRIX/UWS Version 4.1 documentation and provides corrections to these problems.

Appendix A – Problems Resolved Since Last Release
Lists the software and documentation problems that have been resolved in this release.

Appendix B – Changes and New Features: Version 4.1
Discusses the new and changed features in ULTRIX/UWS Version 4.1.

Appendix C – Changes and New Features: Version 4.0
Discusses the new and changed features in ULTRIX/UWS Version 4.0.

Appendix D – RISC-VAX System Differences and Porting Hints
Discusses how to port software from VAX to RISC systems.

Related Documentation

You should have the printed documentation kit for the ULTRIX/UWS Version 4.1 release and your hardware documentation.

Aside from this document, the four documents in the ULTRIX/UWS Version 4.1 release that are most likely to help you get started are:

- *ULTRIX/UWS Version 4.1 Software Product Description*
- *Basic Installation Guide*
- *Advanced Installation Guide*
- *Introduction to System and Network Management*

In addition, you should refer to the *ULTRIX Optional Products Cross Reference Table* for information about the separately-licensed products supported by ULTRIX/UWS Version 4.1.

Text Conventions

The following conventions are used in this document:

%	The default user prompt is your system name followed by a right angle bracket. In this manual, a percent sign (%) is used to represent this prompt.
#	A number sign is the default superuser prompt.
>>> CPU*nn*>>	The console subsystem prompt is two right angle brackets on RISC systems, or three right angle brackets on VAX systems. On a system with more than one central processing unit (CPU), the prompt displays two numbers: the number of the CPU, and the number of the processor slot containing the board for that CPU.
user input	This bold typeface is used in interactive examples to indicate typed user input.
system output	This typeface is used in interactive examples to indicate system output and also in code examples and other screen displays. In text, this typeface is used to indicate the exact name of a

	command, option, partition, pathname, directory, or file.
UPPERCASE lowercase	The ULTRIX system differentiates between lowercase and uppercase characters. Literal strings that appear in text, examples, syntax descriptions, and function definitions must be typed exactly as shown.
rlogin	In syntax descriptions and function definitions, this typeface is used to indicate terms that you must type exactly as shown.
macro	In text, bold type is used to introduce new terms.
filename	In examples, syntax descriptions, and function definitions, italics are used to indicate variable values; and in text, to give references to other documents.
[ ]	In syntax descriptions and function definitions, brackets indicate items that are optional.
{ | }	In syntax descriptions and function definitions, braces enclose lists from which one item must be chosen. Vertical bars are used to separate items.
. . .	In syntax descriptions and function definitions, a horizontal ellipsis indicates that the preceding item can be repeated one or more times.
. . .	A vertical ellipsis indicates that a portion of an example that would normally be present is not shown.
`cat`(1)	Cross-references to the *ULTRIX Reference Pages* include the appropriate section number in parentheses. For example, a reference to `cat`(1) indicates that you can find the material on the `cat` command in Section 1 of the reference pages.
RETURN	This symbol is used in examples to indicate that you must press the named key on the keyboard.
CTRL/*x*	This symbol is used in examples to indicate that you must hold down the CTRL key while pressing the key *x* that follows the slash. When you use this key combination, the system sometimes echoes the resulting character, using a circumflex (^) to represent the CTRL key (for example, ^C for CTRL/C). Sometimes the sequence is not echoed.
ESC X	This symbol is used in examples to indicate that you must press the first named key and then press the second named key. In text, this combination is indicated as ESC-X.
MB1,MB2,MB3	Unless the mouse buttons have been redefined, MB1 indicates the left mouse button, MB2 indicates the middle mouse button, and MB3 indicates the right mouse button.

Installation Notes 1

This chapter discusses issues and known problems with the installation procedure and, when possible, provides solutions or workarounds to the problems. Read this chapter before you install ULTRIX/UWS Version 4.1.

For additional installation notes specific to individual processors, see Chapter 2, Processor-Specific Notes, as well.

The notes in this chapter cover the following topics:

- Hardware
- Boot
- General Installation
- Configuration
- Diskless Management Services (DMS)
- Remote Installation Services (RIS)

1.1 Hardware

This section contains notes about hardware and peripheral devices.

1.1.1 KDB50 ECO Level

If you use KDB50s on a BI-based VAX system, we recommend that customer service ensure that the BIIC chip on the KDB50 is revision level 5 or higher. Otherwise, there is a small chance that machine checks will be generated by the BI bus.

1.1.2 RQDX Q-bus Controller Jumper Settings

If there are multiple RQDX controllers and the RQDX2 is the last controller on the bus, ignore the jumper configuration stated in the hardware manual. The jumper setting should be one (1), not four (4) as stated in the hardware manual.

1.1.3 TK50 and TK70 Tape Usage

When a blank TK50 or TK70 tape is inserted into the drive, calibration marks are written on the tape. These calibration marks determine the tape format. The tape format can only be changed by a bulk tape eraser.

When using TK50 and TK70 tapes, the following rules apply:

- A TK50 formatted tape can be written and read on a TK50 drive.
- A TK50 formatted tape can be read but not written on a TK70 drive. The TK70 drive considers TK50 tapes to be hardware write-protected.
- A TK70 formatted tape can be written or read on a TK70 drive.
- A TK70 tape is completely unusable on a TK50 drive.

Violation of these rules results in command failure and drive error log messages.

1.1.4 Data Corruption from Programs Accessing Tape Units

The TK70, TS11, and TU81 tape units require the data buffers to be aligned on a machine word boundary. Data corruption can occur if the data buffer boundaries are not aligned properly. When the buffer is declared as a local variable, the buffer will be on the user's program stack; alignment is therefore not ensured and may cause data corruption.

To ensure that the data buffer is correctly aligned, all programs that access tapes must declare the data buffer as a type static or as a global variable. The compilers then assure proper alignment of the data buffer and data.

1.1.5 Required Switch Settings for TSV05 Tape Drive

The switch settings for the TSV05 controller (M7196 and M7206) as described in the manuals *TSV05 Tape Transport – Pocket Service Guide* (EK-TSV05-PS-005) and *TSV05 Tape Transport Subsystem – Technical Manual* (EK-TSV05-TM-004) are incorrect.

The factory switch setting information for the M7206-PA module in the manuals lists switch E61-9 OFF. This factory switch setting does not work on systems running ULTRIX Version 3.0 and higher. The tape device always appears off line to the operating system if the switch is set OFF.

1.1.6 Installing from a TE16 Tape Drive

During installation, you are asked to identify the software distribution device. If you are using a TE16 tape drive, choose the TU77 tape drive option. When your system is booted, the TE16 will be identified correctly.

1.1.7 TU81 Tape Drive

A problem in the TU81 tape unit can cause data transfers to fail. All users that have TU81 tape units should contact a customer service representative to ensure that FCO number TU81 R-005 is applied and that the revision level is up to at least D1. Units that do not have this FCO applied will experience hard errors logged and the unit's controller fault light will light.

1.1.8 MSCP Disks Remain Off Line If Switched Off Line While in Use

If a disk unit served by the MSCP driver (any RA disk) is switched off line while operations are in progress, the disk cannot be brought back on line. If this happens, in-progress and subsequent data transfer operations to the unit will fail. The system call that failed will return an EIO error.

This condition can be cleared only by setting the unit back on line and rebooting the system.

1.1.9 HSC Microcode ECO Level for MSCP Disks

Proper operation of the MSCP disk subsystem requires that the HSC software subsystem microcode ECO level should be level Version 3.9A or higher.

1.1.10 No Bad Block Replacement on MASSBUS Disk Media

The VMB boot driver used throughout the new boot path cannot handle bad blocks on the media. Therefore, components of the boot path, superblocks, directories, and `/vmunix`, cannot cross a bad sector on the disk. If they encounter a bad sector on the disk, you will receive fatal controller errors. ULTRIX bad block replacement strategies do not exist in the VMB boot driver.

1.1.11 Eight-Bit Terminal Driver Support

You must set up your hardware and software properly if you intend to use a terminal in full eight-bit mode. Refer to `gettytab`(5) for instructions on how to enable logins on terminal lines that require eight-bit characters. The `p8` and `pd` flags have been added to `gettytab` to facilitate the use of eight-bit characters.

DEC VT100 series terminals are capable of displaying only the lower half of the DEC Multinational Character Set. Standard seven-bit ASCII characters are included in the lower half of the Multinational Character Set and ISO-8859/2.

DEC VT200 and VT300 series terminals are capable of displaying the full DEC Multinational Character Set and ISO-8859/2. However, they do not display eight-bit characters when they are in VT100 mode. To determine the current terminal mode, call up the terminal's Set-Up Directory menu and select the `General` setup option.

For example, to change your VT220 terminal set up into eight-bit mode, follow these steps:

1. Call up the terminal's setup menu by pressing the Setup (F3) key. Select the `General` menu option.
2. Move to the field that allows you to select the terminal mode. Select the option `VT200 mode 8 bit controls.`
3. Select the `To Directory` option to return you to top level.
4. Call up the `Comm` menu. Select the `8 Bits No Parity` option.
5. Exit from setup mode by pressing the Setup key again.

Note that when you change a VT200 or VT300 series terminal from VT100 mode, the F11 key no longer represents the escape key. Refer to your terminal's installation guide for a complete description of terminal setup.

The DECwindows terminal emulator, `dxterm`(1X), can also be set up for use with eight-bit characters. In this case, the terminal mode must be set to VT300 mode, 8-bit control.

1.1.12 Scrambled Stack Printouts on System Console

Kernel and interrupt stacks that are printed at the consoles may become scrambled. If this happens, lower the baud rate of the console.

1.2 Boot

This section contains notes about booting.

1.2.1 Conversational Boot Problem on MSCP-Type Disk

On MSCP-type disk drives, it is possible for the booted drive to go off line during a conversational boot. If you do not supply input to the `Enter image name:` prompt within several minutes, the booted MSCP drive will go off line and subsequent reads will fail. To prevent this problem, enter the image name before the timeout period. Otherwise, a reboot will be required. See the *Guide to System Shutdown and Startup* for information on the conversational boot procedure.

1.3 General Installation

The following notes apply to the installation of ULTRIX/UWS Version 4.1. For additional installation notes specific to individual processors, see the appropriate section under Processor–Specific Notes.

1.3.1 Media Labels for ULTRIX/UWS Version 4.1

Table 1-1 lists the media labels for ULTRIX/UWS Version 4.1.

Table 1-1: ULTRIX/UWS Version 4.1 Media Labels

Media Type (RISC)	Media Label	Media Type (VAX)	Media Label
TK50	ULTRIX/UWS V4.1 (RISC) SUPPORTED VOL 1	TK50	ULTRIX/UWS V4.1 (VAX) SUPPORTED
	ULTRIX/UWS V4.1 (RISC) SUPPORTED VOL 2		ULTRIX/UWS V4.1 (VAX) UNSUPPORTED
	ULTRIX/UWS V4.1 (RISC) MANDATORY UPGRADE		ULTRIX/UWS V4.1 (VAX) MANDATORY UPGRADE
	ULTRIX/SQL V1.1 (RISC)		ULTRIX/SQL V1.1 (VAX)
	ULTRIX/UWS V4.1 (RISC) UNSUPPORTED		

Table 1-1: (continued)

Media Type (RISC)	Media Label	Media Type (VAX)	Media Label
MT9	ULTRIX/UWS V4.1 (RISC) SUPPORTED VOL 1	MT9	ULTRIX/UWS V4.1 (VAX) SUPPORTED VOL 1
	ULTRIX/UWS V4.1 (RISC) SUPPORTED VOL 2		ULTRIX/UWS V4.1 (VAX) SUPPORTED VOL 2
	ULTRIX/UWS V4.1 (RISC) SUPPORTED VOL 3		ULTRIX/UWS V4.1 (VAX) MANDATORY UPGRADE
	ULTRIX/UWS V4.1 (RISC) SUPPORTED VOL 4		ULTRIX/SQL V1.1 (VAX)
	ULTRIX/UWS V4.1 (RISC) MANDATORY UPGRADE		ULTRIX/UWS V4.1 (VAX) UNSUPPORTED
	ULTRIX/SQL V1.1 (RISC)		
	ULTRIX/UWS V4.1 (RISC) UNSUPPORTED		
CDROM	ULTRIX/UWS V4.1 SUPP/UNSUPP (RISC) Includes Mandataory UPG	CDROM	ULTRIX/UWS V4.1 SUPP/UNSUPP (VAX) Includes Mandataory UPG
RA60	–-	RA60	ULTRIX/UWS V4.1 SUPP/UNSUPP (VAX) Includes Mandataory UPG

1.3.2 ULTRIX/UWS Version 4.1 Subset Sizes

The following sections list the subset sizes for the Supported and Unsupported subsets that make up ULTRIX/UWS Version 4.1. For a description of each subset, please refer to the *Advanced Installation Guide*.

1.3.2.1 ULTRIX/UWS Version 4.1 Supported Subsets – The following tables list the supported RISC and VAX subsets that make up ULTRIX/UWS Version 4.1.

Table 1-2 lists the supported RISC ULTRIX subset sizes in kilobytes for the `root`, `/usr`, and `/var` directories.

Table 1-2: ULTRIX Supported Subset Sizes (RISC)

Subset RISC	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
UDTACCT410	0.043	266.240	–-	266.283
UDTAFM410	–-	871.891	–-	871.891
UDTBASE410	4577.730	24073.700	33.506	28684.900
UDTBIN410	12.412	28623.900	–-	28636.300
UDTCOMM410	13.066	1024	–-	1037.070
UDTDCMT410	0.017	1433.870	–-	1433.890

Table 1-2: (continued)

Subset RISC	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
UDTDL410	140.232	4124.110	–-	4264.340
UDTEXER410	–-	1461.670	–-	1461.670
UDTINET410	356.124	5074.090	152.551	5582.770
UDTINTL410	–-	501.734	–-	501.734
UDTKERB410	–-	2715.020	764.416	3479.440
UDTMAN410	–-	4298.660	–-	4298.660
UDTMH410	0.512	7427.400	1.024	7428.940
UDTMOP410	26.043	577.584	139.776	743.403
UDTNFS410	209.514	1426.910	478.299	2114.720
UDTPGMR410	0.016	7344.650	–-	7344.670
UDTPRESTO410	0.034	208.896	–-	208.930
UDTPRINT410	36.362	2955.740	0.512	2992.610
UDTRPCDEV410	–-	607.904	90.112	698.016
UDTRPCRT410	–-	281.795	1191.330	1473.130
UDTSCCS410	–-	1255.780	–-	1255.780
UDTSEC410	81.463	893.658	–-	975.121
UDTSMSCAMP410	–-	99.873	–-	99.873
UDTUMAIL410	28.041	668.830	–-	696.871
UDTUUCP410	0.020	581.632	1031.180	1612.830
TOTALS	5481.630	98799.500	3882.710	108164

Table 1-3 lists the supported RISC UWS subset sizes in kilobytes for the `root`, `/usr`, and `/var` directories.

Table 1-3: UWS Supported Subset Sizes (RISC)

Subset RISC	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
UDWDECW410	–-	12616.200	–-	12616.200
UDWFONT15410	–-	2815.470	–-	2815.470
UDWFONT410	–-	2475.180	–-	2475.180
UDWFONTSTR410	–-	162.304	–-	162.304
UDWMAN410	–-	1468.470	–-	1468.470
UDWSER410	5.120	9062.310	–-	9067.430

Table 1-3: (continued)

Subset RISC	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
UDWX11410	–-	7133.340	–-	7133.340
UDWXDEV410	–-	15274.200	–-	15274.200
TOTALS	5.120	51007.500	–-	51012.600

Table 1-4 lists the supported VAX ULTRIX subset sizes in kilobytes for the `root`, `/usr`, and `/var` directories.

Table 1-4: ULTRIX Supported Subset Sizes (VAX)

Subset VAX	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
ULTACCT410	0.043	145.408	–-	145.451
ULTAFM410	–-	871.891	–-	871.891
ULTBASE410	2796.730	12630.800	33.506	15461.000
ULTBIN410	12.411	6295.840	–-	6308.250
ULTBSC410	0.036	208.896	–-	208.932
ULTCOMM410	13.066	628.736	–-	641.802
ULTDCMT410	0.017	1038.270	–-	1038.290
ULTDL410	81.864	1462.440	–-	1544.300
ULTEXER410	–-	862.629	–-	862.629
ULTINET410	218.908	2971.820	106.471	3297.200
ULTINTL410	–-	286.238	–-	286.238
ULTKERB410	–-	1308.540	538.112	1846.650
ULTMAN410	–-	4365.240	–-	4365.240
ULTMH410	0.512	4500.470	1.024	4502.010
ULTMOP410	26.043	329.776	84.532	440.351
ULTNFS410	115.306	769.502	261.211	1146.020
ULTPASCAL410	–-	721.450	–-	721.450
ULTPGMR410	0.016	3882.280	–-	3882.300
ULTPRESTO410	0.034	119.808	–-	119.842
ULTPRINT410	18.954	2160.210	0.512	2179.680
ULTRPCDEV410	–-	418.032	53.248	471.280
ULTRPCRT410	–-	281.795	739.745	1021.540
ULTSCCS410	–-	732.087	–-	732.087
ULTSEC410	251.447	547.546	–-	798.993
ULTSMSCAMP410	–-	99.873	–-	99.873
ULTUMAIL410	28.041	425.118	–-	453.159

Table 1-4: (continued)

Subset VAX	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
ULTUUCP410	0.020	339.968	630.793	970.781
ULTVAXC410	–-	878.598	–-	878.598
TOTALS	3563.450	49283.300	2449.150	55295.900

Table 1-5 lists the supported VAX UWS subset sizes in kilobytes for the `root`, `/usr`, and `/var` directories.

Table 1-5: UWS Supported Subset Sizes (VAX)

Subset VAX	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
UWSDECW410	–-	8066.570	–-	8066.570
UWSFONT15410	–-	2796.910	–-	2796.910
UWSFONT410	–-	2456.620	–-	2456.620
UWSMAN410	–-	1560.570	–-	1560.570
UWSPSVIEW410	–-	3689.460	–-	3689.460
UWSSER410	5.120	5080.070	–-	5085.190
UWSX11410	–-	4102.310	–-	4102.310
UWSXDEV410	–-	5990.430	–-	5990.430
TOTALS	5.120	37665.500	–-	37670.700

1.3.2.2 ULTRIX/UWS Version 4.1 Mandatory Upgrade Subset Sizes – The following tables list the RISC and VAX subsets that make up the ULTRIX/UWS Version 4.1 Mandatory Upgrade.

Table 1-6 lists the RISC ULTRIX Mandatory Upgrade subset sizes in kilobytes for the `root`, `/usr`, and `/var` directories.

Table 1-6: ULTRIX Mandatory Upgrade Subset Sizes (RISC)

Subset RISC	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
UDTBASE402	5167.410	9073.750	22.842	14264
UDTBIN402	–-	28523.800	–-	28523.800
UDTCOMM402	–-	172.032	–-	172.032
UDTDL402	–-	4088.260	–-	4088.260
UDTEXER402	–-	77.824	–-	77.824

Table 1-6: (continued)

Subset RISC	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
UDTINET402	–-	1528.380	–-	1528.380
UDTINTL402	–-	16.437	–-	16.437
UDTKERB402	–-	18.299	–-	18.299
UDTMAN402	–-	4300.030	–-	4300.030
UDTMH402	–-	1045.300	–-	1045.300
UDTMOP402	25.927	499.712	–-	525.639
UDTNFS402	–-	184.320	–-	184.320
UDTPGMR402	–-	2317.060	–-	2317.060
UDTPRESTO402	–-	1479.680	0.539	1480.220
UDTPRINT402	–-	446.464	–-	446.464
UDTSMSCAMP402	–-	99.873	–-	99.873
UDTUMAIL402	–-	270.336	–-	270.336
TOTALS	5193.340	54141.600	23.381	59358.300

Table 1-7 lists the RISC UWS Mandatory Upgrade subset sizes in kilobytes for the `root`, `/usr`, and `/var` directories.

Table 1-7: UWS Mandatory Upgrade Subset Sizes (RISC)

Subset RISC	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
UDWDECW402	–-	12616.200	–-	12616.200
UDWMAN402	–-	1468.470	–-	1468.470
UDWSER402	3.584	9091.510	–-	9095.090
UDWX11402	–-	7133.330	–-	7133.330
UDWXDEV402	–-	15274	–-	15274
TOTALS	3.584	45583.500	–-	45587.100

Table 1-8 lists the VAX ULTRIX Mandatory Upgrade subset sizes in kilobytes for the `root`, `/usr`, and `/var` directories.

Table 1-8: ULTRIX Mandatory Upgrade Subset Sizes (VAX)

Subset VAX	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
ULTBASE402	2258.770	5413.640	22.842	7695.250
ULTBIN402	–-	6241.230	–-	6241.230
ULTCOMM402	–-	106.496	–-	106.496
ULTDL402	–-	1429.140	–-	1429.140
ULTEXER402	–-	17.408	–-	17.408
ULTINET402	–-	1146.430	–-	1146.430
ULTINTL402	–-	16.437	–-	16.437
ULTKERB402	–-	18.299	–-	18.299
ULTMAN402	–-	4427.420	–-	4427.420
ULTMH402	–-	631.808	–-	631.808
ULTMOP402	25.927	282.624	–-	308.551
ULTNFS402	–-	113.664	–-	113.664
ULTPGMR402	–-	1490.090	–-	1490.090
ULTPRESTO402	–-	119.808	–-	119.808
ULTPRINT402	–-	269.312	–-	269.312
ULTSMSCAMP402	–-	99.873	–-	99.873
ULTUMAIL402	–-	145.408	–-	145.408
TOTALS	2284.700	21969.100	22.842	24276.600

Table 1-9 lists the VAX UWS Mandatory Upgrade subset sizes in kilobytes for the `root`, `/usr`, and `/var` directories.

Table 1-9: UWS Mandatory Upgrade Subset Sizes (VAX)

Subset VAX	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
UWSDECW402	–-	7846.410	–-	7846.410
UWSMAN402	–-	1560.570	–-	1560.570
UWSSER402	5.120	5111.310	–-	5116.430
UWSX11402	–-	4102.320	–-	4102.320
UWSXDEV402	–-	5990.380	–-	5990.380
TOTALS	5.120	24611	–-	24616.100

Table 1-10 lists the RISC SQL subset sizes in kilobytes for the `root`, `/usr`, and `/var` directories.

Table 1-10: SQL Subset Sizes (RISC)

Subset RISC	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
QLRBASE110	–	26874.800	1510.140	28384.900
QLRDEVT110	–	6622.340	–	6622.340
QLRERRMSG110	–	745.951	–	745.951
QLRMAN110	–	86.337	–	86.337
TOTALS	–	34329.4	1510.140	35839.600

Table 1-11 lists the VAX SQL subset sizes in kilobytes for the `root`, `/usr`, and `/var` directories.

Table 1-11: SQL Subset Sizes (VAX)

Subset VAX	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
QLVBASE110	–-	12856.700	1560.800	14417.500
QLVDEVT110	–-	1847.520	–-	1847.520
QLVERRMSG110	–-	745.951	–-	745.951
QLVMAN110	–-	86.337	–-	86.337
TOTALS	–-	15536.500	1560.800	17097.300

1.3.2.3 ULTRIX/UWS Version 4.1 Unsupported Subsets – The following tables list the unsupported subsets in ULTRIX/UWS Version 4.1.

Table 1-12 lists the RISC ULTRIX unsupported subset sizes in kilobytes for the `root`, `/usr`, and `/var` directories.

Table 1-12: ULTRIX Unsupported Subset Sizes (RISC)

Subset RISC	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
UDXBASE410	94.400	3511.820	0.512	3606.730
UDXBIB410	–-	291.745	–-	291.745
UDXCOURIER410	–-	164.438	–-	164.438
UDXDOC410	–-	3447.050	–-	3447.050

Table 1-12: (continued)

Subset RISC	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
UDXEDIT410	–-	6266.650	–-	6266.650
UDXGAMES410	–-	2510.780	–-	2510.780
UDXLEARN410	–-	733.757	–-	733.757
UDXMAN410	–-	128.685	–-	128.685
UDXNEWS410	–-	1310.500	–-	1310.500
UDXNOTES410	–-	1884.920	–-	1884.920
UDXRCS410	–-	212.539	–-	212.539
UDXSHELLS410	–-	94.160	–-	94.160
UDXTOOLS410	–-	111.554	–-	111.554
TOTALS	94.400	20668.600	0.512	20763.500

Table 1-13 lists the RISC UWS unsupported subset sizes in kilobytes for the `root`, `/usr`, and `/var` directories.

Table 1-13: UWS Unsupported Subset Sizes (RISC)

Subset RISC	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
UDXUNCOMP410	–-	1690.300	–-	1690.300
UDXUNMAN410	–-	148.501	–-	148.501
UDXUNMIT410	–-	16454.800	–-	16454.800
TOTALS	–-	18293.600	–-	18293.600

Table 1-14 lists the VAX ULTRIX unsupported subset sizes in kilobytes for the `root`, `/usr`, and `/var` directories.

Table 1-14: ULTRIX Unsupported Subset Sizes (VAX)

Subset VAX	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
ULXAPL410	–-	269.805	–-	269.805
ULXBASE410	55.783	2285.560	0.512	2341.860
ULXBIB410	–-	194.465	–-	194.465
ULXCOURIER410	–-	103.618	–-	103.618
ULXCPM410	–-	28.934	–-	28.934
ULXDOC410	–-	3447.050	–-	3447.050

Table 1-14: (continued)

Subset VAX	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
ULXEDIT410	––	6266.650	––	6266.650
ULXF77410	––	732.498	––	732.498
ULXGAMES410	––	2232.120	––	2232.120
ULXHYPER410	––	78.723	––	78.723
ULXICON410	––	346.396	––	346.396
ULXINGRES410	––	2637.600	––	2637.600
ULXLEARN410	––	652.284	––	652.284
ULXLISP410	––	3962.890	––	3962.890
ULXMAN410	––	318.958	––	318.958
ULXMOD2410	––	1035.540	––	1035.540
ULXNEWS410	––	1310.500	––	1310.500
ULXNOTES410	––	1178.360	––	1178.360
ULXRCS410	––	226.655	––	226.655
ULXSHELLS410	––	55.248	––	55.248
ULXSPMS410	––	1238.100	––	1238.100
ULXTOOLS410	––	54.210	––	54.210
ULXVARIAN410	––	2735.770	––	2735.770
TOTALS	55.783	31391.900	0.512	31448.200

Table 1-15 lists the VAX UWS unsupported subset sizes in kilobytes for the `root`, `/usr`, and `/var` directories.

Table 1-15: UWS Unsupported Subset Sizes (VAX)

Subset VAX	root size (kbytes)	/usr size (kbytes)	/var size (kbytes)	Total (kbytes)
ULXUNCOMP410	––	1514.170	––	1514.170
ULXUNMAN410	––	148.501	––	148.501
ULXUNMIT410	––	9457.380	––	9457.380
TOTALS	––	11120.100	––	11120.100

1.3.3 Time Set During System Installation May Be Incorrect for GMT Offsets

During the installation of ULTRIX/UWS Version 4.1, you are asked to specify the current date, time, and time zone. If you specify the time zone as a number of hours east of Greenwitch Mean Time (GMT) and specify that your area does not alternate

between Daylight Savings Time and Standard Time, the ULTRIX operating system will set its clock one hour earlier than the time you specified.

To work around this problem, reset your local time manually using the `date(1)` command after you complete your installation.

1.3.4 Installing Layered Products and Unsupported Software

Some layered products and unsupported software expect ULTRIX/UWS Version 4.0 lock files to be present on the system when they are installed. Because ULTRIX/UWS Version 4.1 software contains no ULTRIX/UWS Version 4.0 lock files, you must create the lock files before you install layered products or unsupported software.

Follow these steps to create the lock files:

1. Log in as `root` or become superuser.
2. Change to the `/usr/etc/subsets` directory as follows:

```
# cd /usr/etc/subsets
```

3. Issue the following commands to create a ULTRIX/UWS Version 4.0 subset lock file for each ULTRIX/UWS Version 4.1 subset you have installed and to set the correct permissions on the files:

```
# foreach i (U*410.lk)
? touch ‘ echo $i | sed -e ’s/410/400/’ ‘
? touch ‘ echo $i | sed -e ’s/410/400/’ | sed -e ’s/lk$/ctrl/’ ‘
? touch ‘ echo $i | sed -e ’s/410/400/’ | sed -e ’s/lk$/scp/’ ‘
? end
# chmod 744 U*400.scp
```

Once you create the ULTRIX/UWS Version 4.0 subset lock files, you can install layered products or unsupported software. For instructions on installing layered products, see the installation guide for the layered product. For instructions on installing unsupported software, see the *Advanced Installation Guide*.

1.3.5 Installing the Mandatory Upgrade

The installation of the Mandatory Upgrade is now supported in the installation of ULTRIX/UWS Version 4.1. This section explains how to install the Mandatory Upgrade if you are installing ULTRIX/UWS Version 4.1 from the following distribution devices:

- TK50 Tape
- 9-track Magnetic Tape (MT9)
- CDROM Optical Disk
- RA60 Disk Pack
- Network

Note

If you are upgrading your Remote Installation Server (RIS) area or upgrading your Diskless Server environments, refer to the *Mandatory Upgrade Installation Instructions* after you have installed the ULTRIX operating system on your server.

1.3.5.1 Installing the Mandatory Upgrade from TK50 or MT9 Tape – During the installation of ULTRIX/UWS Version 4.1, you are prompted with the following message:

```
Do you have a Mandatory Upgrade to be installed at this time? (y/n) [n]:
```

Answer yes to the prompt.

You are then asked to remove the tape from the tape drive and load the tape with the word UPGRADE on its label into the tape drive.

```
Please mount the tape containing the Mandatory Upgrade subsets.
```

After the tape is loaded, you are asked if the tape is ready and online.

```
Please make sure your installation tape is mounted and online.
```

When the tape is mounted and online, the system begins loading the Mandatory Upgrade subsets depending on whether you are performing a basic or an advanced installation.

- Basic Installation

  If you are performing a basic installation, the standard Mandatory Upgrade subsets are automatically installed.

- Advanced Installation

  If you are performing an advanced installation, the standard Mandatory Upgrade subsets are automatically installed and you are given a menu from which you may choose to install Mandatory Upgrade subsets for optional subsets.

```
*** SUPPORTED SOFTWARE INSTALLATION ***

*** Enter Subset Selections ***

The following subsets are mandatory and will be installed automatically:
 * Base System UPGRADE                  * Kernel Config Files UPGRADE           * T(
 * X11/DECwindows User Envrment UPGRADE

The subsets listed below are optional:
1) On Line Manual Pages UPGRADE        2) Prestoserve Utilities UPGRADE
3) Additional DECwindows Appl UPGRADE  4) UWS References Pages UPGRADE
5) Worksystem Development S/W UPGRADE

6) All of the Above
7) None of the Above
8) Exit without installing subsets

Enter your choice(s):
After you make your selection, you are prompted to confirm
your choices.

Is this correct? (y/n):
```

If you answer no, the system reprints the menu and you may select the optional Mandatory Upgrade subsets again. If you answer yes, the system installs the upgrade subsets.

If for some reason the system is unable to install the mandatory upgrade subsets, you will receive the following error message:

```
The installation procedure was unable to install
the Mandatory Upgrade subsets.
```

The system will then attempt to install the Mandatory Upgrade subsets a second time.

```
*** Attempting again to install the Mandatory Upgrade subsets ***
```

If the system is still unable to install the mandatory upgrade subsets, you will receive the following error message:

```
The installation procedure failed to install the Mandatory Upgrade subsets.
This causes the installation procedure to stop.
Contact your DIGITAL representative.
```

1.3.5.2 Installing from an RA60 or CDROM Disk – If you are installing from an RA60 or CDROM Disk, the standard mandatory upgrade subsets will be automatically installed during the installation of ULTRIX/UWS Version 4.1 and the system will display the following message:

```
Installing Mandatory Upgrade
```

If for some reason the system is unable to install the mandatory upgrade subsets, you will receive the following error message:

```
The installation procedure was unable to install
the Mandatory Upgrade subsets.
```

The system will then attempt to install the Mandatory Upgrade subsets a second time.

```
*** Attempting again to install the Mandatory Upgrade subsets ***
```

If the system is still unable to install the mandatory upgrade subsets, you will receive the following error message:

```
The installation procedure failed to install the Mandatory Upgrade subsets.
This causes the installation procedure to stop.
Contact your DIGITAL representative.
```

1.3.5.3 Installing from the Network – If you are installing ULTRIX/UWS Version 4.1 from the network, the system will begin loading the Mandatory Upgrade subsets based on the whether you are performing a basic or an advanced installation.

- Basic Installation

  If you are performing a basic installation from the network, the standard Mandatory Upgrade subsets are automatically installed.

- Advanced Installation

  If you are performing an advanced installation, the standard Mandatory Upgrade subsets are automatically installed and you are given a menu from which you may choose to install Mandatory Upgrade subsets for optional subsets.

```
After you make your selection, you are prompted to confirm
your choices.


*** SUPPORTED SOFTWARE INSTALLATION ***


*** Enter Subset Selections ***

The following subsets are mandatory and will be installed automatically:
 * Base System                          * Kernel Configuration Files
 * TCP/IP Networking Utilities          * Network File System Utilities
 * Extended (Berkeley) Mailer           * Base System UPGRADE
 * Kernel Config Files UPGRADE          * TCP/IP Networking Utilities UPGRADE
 * X11/DECwindows Servers               * X11/DECwindows User Environment
 * X11/DECwindows 75dpi Fonts           * X11/DECwindows Servers UPGRADE
 * X11/DECwindows User Envrment UPGRADE

The subsets listed below are optional:
 1) System Exerciser Package             2) RAND Mail Handler
 3) Kerberos Network Authentication       4) Enhanced Security Features
 5) SCAMP sys config and mgmt program     6) Prestoserve Utilities
 7) Document Preparation Software         8) Printer Support Environment
 9) Adobe Font Metric Files              10) Software Development Utilities
11) RPC Runtime Environment              12) RPC Development Environment
13) Internationalization Tools           14) Source Code Control System
15) Pascal Development Package           16) VAX C/ULTRIX
17) On Line Manual Pages                 18) Accounting Software
19) Communications Utilities             20) Bisynchronous Communications
21) Maintenance Operations Protocol      22) Unix-to-Unix Copy Facility
23) On Line Manual Pages UPGRADE         24) Prestoserve Utilities UPGRADE
25) X11/DECwindows 100dpi Fonts          26) VS35XX X11/DECwindows Fonts
27) Additional DECwindows Applications   28) Worksystem Development Software
29) UWS Reference Pages                  30) Additional DECwindows Appl UPGRADE
31) UWS References Pages UPGRADE         32) Worksystem Development S/W UPGRADE

33) All of the Above
34) None of the Above
35) Exit without installing subsets

Enter your choice(s):
```

If you answer no, the system reprints the menu and you may select the optional Mandatory Upgrade subsets again. If you answer yes, the system installs the upgrade subsets.

1.3.6 Removing Subsets

ULTRIX/UWS Version 4.1 comes with several thousand files, which are organized into discrete functional units, called subsets. If you need to remove any of the files that were placed on your system when you installed ULTRIX/UWS Version 4.1, use the `setld` utility with the `-d` option. Failure to use the `setld` command may degrade the usability of the software.

The `setld` utility tracks the files installed on your system by using data that it stores in `/usr/etc/subsets`. Do not remove any of the files in that directory, because you will lose the ability to install and delete system and layered product software.

For more information on the `setld` utility, see the *ULTRIX Reference Pages*.

1.3.7 Unsupported Subsets

To install the unsupported subsets from tape, load the unsupported tape. To install subsets from your CDROM or RA60 distribution, mount the `c` partition of the CDROM on /mnt and load subsets from `/mnt/RISC/UNSUPPORTED` or `/mnt/VAX/UNSUPPORTED`, as is appropriate.

Note

The unsupported software subset descriptions, including sizes and dependencies, can be found in the *Advanced Installation Guide*.

1.4 Configuration

This section contains notes about system configuration.

1.4.1 Secure Console

Even a system that is running in secure console mode can be interrupted during a reboot and brought up in single-user mode. Thus, if your system is running in secure console mode, the system administrator should edit the `/etc/rc` file to prevent reboots from being interrupted.

To edit the `/etc/rc` file to prevent reboots from being interrupted, follow these steps:

1. Add the following line to the top of the `/etc/rc` file. This line should be the first line in the file.

```
trap '' 1 2 3
```

2. Find the case statement in the `/etc/rc` file that reboots the system after `fsck(8)` is performed on the file system. The case statement looks like this:

```
echo Automatic reboot in progress... >/dev/console
/etc/fsck -p >/dev/console
case $? in

    0)
            ;;
    4)
            /etc/reboot -n
            ;;
    8)
            echo "Automatic reboot failed... help!" >/dev/console
            exit 1
            ;;
   12)
            echo "Reboot interrupted" >/dev/console
            exit 1
            ;;
    *)
            echo "Unknown error in reboot" > /dev/console
            exit 1
            ;;
esac
```

3. Change all occurrences of `exit 1` in the preceding case statement to `/etc/halt.` Making this change will cause the system to shut down again if there are any attempts to gain access to the system by interrupting the boot.

After you have edited the preceding case statement, it should look like this:

```
echo Automatic reboot in progress... >/dev/console
/etc/fsck -p >/dev/console
case $? in

    0)
            ;;
    4)
            /etc/reboot -n
            ;;
    8)
            echo "Automatic reboot failed... help!" >/dev/console
            /etc/halt
            ;;
    12)
            echo "Reboot interrupted" >/dev/console
            /etc/halt
            ;;
    *)
            echo "Unknown error in reboot" > /dev/console
            /etc/halt
            ;;
esac
```

If a system problem arises that warrants booting to single-user mode, you can change the console secure mode and boot the system to single-user mode at the console command line as follows:

```
>> boot -s
```

1.4.2 Dump Device

If your boot device is an RA disk and the dump device is also an RA disk, both disks must reside on the same controller because the disk class driver uses VMB to dump core. In the past, these device drivers contained part of the special design dump code; this is no longer true.

1.4.3 Configuration of Q-bus Terminal Multiplexer Lines

The installation process creates 8 terminal lines for each `cxa16` or `cxb16` terminal multiplexer and 16 lines for the `cxf32` terminal multiplexer. The correct number of lines is 16 for a `cxa16` or `cxb16` and 32 lines for a `cxf32`.

To correct this problem, remake the lines as follows:

1. Log in as `root` or become superuser.
2. Remove the `/dev/tty` lines that correspond to the `cxa16`, `cxb16`, or `cxf32` multiplexers by using the `rm` command. You can identify the lines that need to be removed by using the `ls -l` command and looking for the major number 33, as follows:

   ```
   # ls -l /dev/tty* | grep 33,
   ```

Note

The `dhv`, `dhq`, and `cxy08` also share the major number, 33. If you delete these lines, they will also have to be remade.

3. Remake the correct number of terminal lines with the `MAKEDEV`(8) command.

   For example, if your system has only two `cxa16` devices, type the following command in the `/dev` directory:

   ```
   # MAKEDEV cxa0 cxa1
   ```

   The `cxf32` is configured as two `cxa16` devices. If your system has a `cxf32`, type the following command in the `/dev` directory:

   ```
   # MAKEDEV cxa0 cxa1
   ```

4. After you create the correct number of lines, update the `/etc/ttys` file to include the previously missing lines. For more information on updating the `/etc/ttys` file, see the *Guide to System Environment Setup*.

1.4.4 System Configuration When Disk Controllers Are in Floating Address Space

When performing system configuration for systems that have UNIBUS/QBUS ADAPTERS with devices in floating address space, the following will now occur.

If the device in floating address space is a disk controller, sizer will no longer assume that any drives are attached to it. The reason for this is that sizer cannot correctly size the number of drives attached to this disk controller in floating address space.

Here is an example of the incorrect configuration file:

```
controlleruda0 at   uba0
controlleruq0  at   uda0 csr 0172150  vector  uqintr
disk ra0  at   uq0  drive 0
controlleruda1 at   uba1
controlleruq17 at   uda1 csr 0160334  vector  uqintr
disk ra1  at   uq17 drive 0
disk ra2  at   uq17 drive 1
disk ra3  at   uq17 drive 2
disk ra5  at   uq17 drive 3
```

In this example we see a disk controller "uq17" at "uda1" in floating address space. As we can see, the old sizer utility assumes that four disk drives are attached to this controller. For the KFQSA controller, this is a wrong assumption by sizer. Since sizer cannot correctly determine the number of drives connected to this disk controller, it will assume none.

Here is an example of the correct configuration file:

```
controlleruda0 at   uba0
controlleruq0  at   uda0 csr 0172150  vector  uqintr
disk ra0  at   uq0  drive 0
controlleruda1 at   uba1
controlleruq17 at   uda1 csr 0160334  vector  uqintr
```

In this example we see a disk controller "uq17" at "uda1" in floating address space. However, sizer now assumes that no drives are attached to it.

Note that this will require the user to edit the configuration file during the installation process to add the correct number of drives for disk controllers that exist in floating address space.

Select the ADVANCED installation option during installation, and when the system asks you if you want to edit the configuration file, type `y.`

Then, do the following:

1. For all disk controllers in floating address space add the correct number of `ra` and `rz` drives attached to it.
2. Make the `ra` and `rz` numbers sequential.

Note

The drive number is the same as the unit plug number on the front panel of the system used to identify the drive. If you are unable to determine what the correct drive numbers are, contact customer service.

1.4.5 Conformance to CSR Address Space Conventions

When you install the ULTRIX software, the UNIBUS and Q-bus devices that reside in floating address space are automatically sized. Therefore, the placement of devices in this space is critical to the success of your installation.

If your configuration does not conform to CSR address assignment rules, you must do an advanced installation. The advanced installation allows you to edit the system configuration file built by the installation software. If you have questions about the accuracy of your floating address space configuration, contact customer service.

1.4.6 The Console Entry in the ttys File

The default terminal type for the console entry in the `/etc/ttys` file is set to VT100. This works if you have a CRT console. However, if you have a hardcopy console, you must change the entry in the `ttys` file to match your console terminal type.

1.4.7 Synchronization Errors for Autodial Modem on a DMF32 Interface

If you have an autodial modem connected to a DMF32 interface and you are using the generic dialer routines in `acucap, tip,` or `uucp,` the system may not be able to open the modem and may print either of the following messages:

```
tip: can't synchronize
uucp: can't synchronize
```

The problem occurs because the DMF32 interface cannot return characters to the system until a carrier is detected by the modem.

If you encounter this problem, edit each entry in `/etc/acucap` that refers to a modem connected to a DMF32 interface to include the `si` Boolean flag. The `si` Boolean flag disables checking of responses from the modem until the carrier is detected. For more information, see `acucap`(5) in the *ULTRIX Reference Pages*.

1.4.8 Shared Lines Do Not Work over Direct Connections

The use of shared lines requires modem control. Carrier detection must be raised upon receipt of an incoming connection and must be dropped when the remote party hangs up. Direct connects that use modem eliminators do not obey this protocol and cannot be used for shared lines. If you try to use shared lines on a direct connect line that has Carrier Detect strapped high, you will disable the line.

1.4.9 Reactivating Hardwired Terminals

Hardwired terminal ports other than the console port may hang as a result of electrical noise appearing on the line when the terminal is turned off and then turned back on. When the port is hung, the terminal does not respond to keyboard input. To reactivate the terminal line, follow these steps:

1. Log in to the console as `root` or become superuser.
2. Determine the number of the hung terminal line by entering the `last` command with the user's login name as an argument.
3. Determine the process identification (PID) of the `getty` process associated with the hung terminal line by entering the `ps` command with the `-ax` option.
4. Use the `kill` command with the `-9` option to kill the process.

The following example shows how to reactivate a hung terminal line. Assume that the login name of the user is kafka:

```
# last kafka
kafka   tty03     Mon Nov 18 10:00   still logged in
kafka   tty03     Mon Nov 18 08:35 - 09:08  (00:33)
kafka   tty03     Mon Nov 18 00:26 - 02:00  (01:33)

# ps -ax

  PID TT STAT  TIME COMMAND
    0 ?  D     0:01 swapper
    1 ?  I     0:34 init
    2 ?  D     0:00 pagedaemon

  159 03 I     0:00 — 2 tty03 (getty)
  160 04 I     0:00 — 2 tty04 (getty)

# kill —9 159
```

1.4.10 Terminals Should Be Left Powered On

Improperly terminated terminal lines can cause the associated `getty` process to use the CPU heavily. Line interference causes the `getty` process to assume that a user is attempting to log in. This problem will be repeated continuously, causing degradation in system performance. Either keep your terminals powered on at all times, or if a terminal line is not used, specify it as "off" in the `/etc/ttys` file to prevent a `getty` process from being started on the line.

1.4.11 Changes to Swap Space and Program Size Parameters

It is now possible to set the maximum data and stack size that a program may grow without affecting the swap configuration parameters.

A few new configurable parameters are added for dynamic swap. The default values, and how to use them in the configuration file, are discussed in the *Guide to Configuration File Maintenance*.

The configuration parameters `dmmin` and `dmmax` are no longer supported. Anyone who uses these configuration variables to increase the data/stack segment sizes is advised to use the new configuration parameters `maxdsiz` and `maxssiz`.

1.4.12 Configuration File Options for Audit

The configuration file AUDIT option loads the optional audit subsystem files into the kernel. The base size of the system audit buffers is described as being configurable at build time by specifying AUDIT=xxx, where xxx is a size. Note that this size is not configurable. The audit subsystem is always configured with its default system audit buffer size.

There is no workaround. Default system audit buffer size cannot be adjusted.

1.4.13 Tuning File System Performance

You can modify the following three system parameters to improve ULTRIX file system performance and demonstrate that improvement in certain types of tests, such as single-process, single-file, cache-sensitive benchmarks:

- Buffer cache size
- ULTRIX write-back scheduling
- The `update` daemon time interval

However, before modifying these parameters, you should be knowledgeable about the ULTRIX operating system. Individual users should analyze their needs by varying the values for each parameter and measuring the effect on performance. Optimal values will differ between workstations, file servers, and timesharing systems.

Note

Unless you understand the value of modifying these parameters and can detect a performance improvement after doing so, you should use their default values.

The following sections describe how to modify each of the parameters.

1.4.13.1 The bufcache Configuration File Parameter – A new configuration file parameter, *bufcache* , allows a specified percentage of physical memory to be set aside by the file system for use by the file system buffer cache. The percentage must be 10 or greater, but less than 100.

By default, buffer cache occupies 10% of main memory. Increasing the buffer cache size means that more file system data is stored in memory. While a large buffer cache may make a benchmark test run faster, there are tradeoffs. The ULTRIX operating system uses a static buffer cache allocation methodology. Main memory that is allocated at boot time for the file system buffer cache cannot be used for user program text or data. Therefore, actual performance depends on the application.

For example, to set the cache buffer size to 25% of memory, add the following line to your system's configuration file located in the directory `/sys/conf/mips` for RISC processors or `/sys/conf/vax` for VAX processors:

```
bufcache              25
```

After editing the configuration file, you need to rebuild your kernel. See the *Guide to Configuration File Maintenance* for more information on the configuration file and its options and for instructions on rebuilding your kernel.

Optimal values for `bufcache` will differ among large timesharing systems, mid-range file servers, and workstations. However, you should not alter `bufcache` if you have a workstation with 8 megabytes of memory. Workstations with 16 megabytes of memory should have a value of no more than 30. If you specify a value greater than 30, your system's file system performance may suffer because of excessive paging and swapping.

For file servers, increasing the buffer cache can improve performance. Note that if you make the buffer cache too large, the resulting system may be less efficient in processing the requests to it from multiple users. To help you determine the optimal value, use the results from the `bufstats` command of the `crash` utility. This command can provide useful data on cache hit/miss ratios. See `crash`(8) in the *ULTRIX Reference Pages* for more information on `bufstats`.

1.4.13.2 ULTRIX Write-Back Scheduling – By default, the ULTRIX operating system returns write requests immediately. If the last byte of a block is written, then the dirty block is asynchronously sent to disk. When this happens, the block becomes unavailable until the disk write completes. While this scheduling method is beneficial in a time-sharing environment, it hinders some benchmark tests which read data immediately after writing it.

To set the ULTRIX system so that data can be read as soon as it is written and writes to disk are delayed as long as possible, make the following change in the `param.c` file located in the directory `/sys/conf/mips` for RISC processors or `/sys/conf/vax` for VAX processors:

```
int delay_wbuffers = 1;
```

After editing the `param.c` file, you need to rebuild the kernel. See the *Guide to Configuration File Maintenance* for instructions on rebuilding your system's kernel.

1.4.13.3 The update Daemon Time Interval – By default, the `update` daemon synchronizes dirty blocks to disk every 30 seconds. You can alter this time interval in two ways. The first way is to add a value to the `/etc/update` command in the file `/etc/rc`. For example, to adjust the `update` time interval from 30 seconds to 2 minutes, edit the file as follows:

```
/etc/update 120;  echo -n update'  >/dev/console
```

The second way is to kill the `update` daemon process and restart it with the new value.

If you have a big cache and an application which often writes over the same blocks of a file, you should consider increasing the time interval for `update`.

1.5 Diskless Management Services (DMS)

This section contains notes about Diskless Management Services (DMS).

1.5.1 DMS Servers and Reference Page Permissions

The permissions and ownership on the ULTRIX and UWS reference pages that are installed on DMS server areas are incorrect and must be set manually.

However, if you install the ULTRIX or UWS reference page subsets on the DMS area of your diskless server and you do not have the corresponding reference page subsets installed on your server system, the script which checks and sets permissions on the ULTRIX/UWS Version 4.1 reference page files will fail with the following error messages:

```
.
.
.

Copying UWS Reference Pages from tape
Verifying UWS Reference Pages (UDWMAN410)

Copying Worksystem Development S/W (UDWXDEV410) from tape

Verifying Worksystem Development S/W (UDWXDEV410)
chown: can't access [A-W]*: No such file or directory
chgrp: access [A-W]*: No such file or directory
chmod: can't access [A-W]*: No such file or directory
chown: can't access [YZ_]*: No such file or directory
chgrp: access [YZ_]*: No such file or directory
chmod: can't access [YZ_]*: No such file or directory
chown: can't access X[A-Z]*: No such file or directory
chgrp: access X[A-Z]*: No such file or directory
chmod: can't access X[A-Z]*: No such file or directory
chown: can't access X[a-z]*: No such file or directory
chgrp: access X[a-z]*: No such file or directory
chmod: can't access X[a-z]*: No such file or directory
.
.
.
```

These errors do not affect the installation of the reference page subsets.

After the installation has completed, whether you receive the error messages or not, you must set the correct ownership and permissions on the the reference page files that were installed in the DMS area on your diskless server by following these steps:

1. Log in as root or become superuser.
2. Change to the directory in the DMS area of your diskless server where the reference pages are installed. On most diskless servers, this area is either /dlenv0/root.mips/usr/man for RISC DMS client areas or /dlenv0/root.vax/usr/man for VAX DMS client areas.
3. Enter the following commands:

```
# find . -type f -exec chmod 444 {} ;
# find . -type f -exec chown root {} ;
# find . -type f -exec chgrp system {} ;
```

1.5.2 Diskless Clients Cannot Update the apropos Index or Use catman

The `apropos` index (`/usr/lib/whatis`) that ships with ULTRIX/UWS Version 4.1 contains entries for all the supported ULTRIX base system reference pages. When additional reference pages are loaded onto the `DMS` client (for example, reference pages that are contained in ULTRIX/SQL subsets), diskless clients cannot access them with the `man` command if the optional `/usr/man/cat?` directories exist.

If a diskless client attempts to use the `man` or `catman` command to update the `apropros` reference page index in `/usr/lib/whatis` or to add preformatted reference pages in the optional `/usr/man/cat?` directories, the command fails with the following message:

```
/bin/sh: /usr/lib/whatis: cannot create
chmod: can't change /usr/lib/whatis: Restricted operation on file system
```

It is possible to work around this limitation as follows:

1. Log in as `root` or become superuser.
2. On the `DMS` client in each area where new reference pages were added, unmount the `/usr` file system, which is normally mounted read-only.
3. On the `DMS` server, temporarily export the `/usr` file system to the `DMS` client read-write from the `/etc/exports` file.
4. On the `DMS` client, re-mount the `/usr` file system read-write and then run the `catman` command with the arguments appropriate for your system.
5. When the `catman` command completes, unmount the `/usr` file system from the client, remove the temporary read-write export line from the `DMS` server's `/etc/exports` file, and remount the `/usr` file system on the client in the normal read-only fashion.

After you complete these steps, `DMS` clients which share the same `/usr` area will have access to the `apropos` index and the reformatted reference pages.

1.5.3 DMS Clients

Diskless Management Services clients should only be registered with one DMS server at a time. Registering with more than one server causes unpredictable results.

1.5.4 Diskless Support

If you install ULTRIX/UWS Version 4.1 in a diskless server area using the `DMS` utility, the diskless server area must have at least ULTRIX/UWS Version 4.0 installed.

Note

If you install ULTRIX/UWS Version 4.1 in a diskless server area, fonts on ULTRIX/UWS Version 4.1 will be incompatible with the X servers from ULTRIX/UWS Version 4.0.

1.5.5 DMS Server with pre-ULTRIX/UWS Version 4.0 Clients

These notes relate to operations with an ULTRIX/UWS Version 4.1 DMS Server serving pre-ULTRIX Version 4.0 clients.

1.5.5.1 DMS Clients and /etc/exports Semantics – The semantics of the `/etc/exports` file changed between ULTRIX-32 Version 3.1 and ULTRIX/UWS Version 4.0. This change affects DMS clients since they are NFS mounted from their server.

If a ULTRIX Worksystem Software Version 2.2 or earlier client is being served from a ULTRIX/UWS Version 4.1 server, the `/etc/exports` entry for the client must be manually updated to ensure that the client gets the same access rights. ULTRIX/UWS Version 4.1 clients automatically make the appropriate changes to a ULTRIX/UWS Version 4.1 server.

Export options are now applied on a per-directory basis, and are no longer inherited from the exported parent file system. As a consequence, you no longer need to export an entire file system in order to export subdirectories within it. However, you do need to be more explicit in specifying options for each exported resource: file system or directory.

The major consequence of this is that to preserve the same behavior on your NFS server as before, you need to apply the export options of the exported parent file system to each of the exported subdirectories. See the note in Appendix C, New `/etc/export` Semantics, for more details on these changes.

Before ULTRIX/UWS Version 4.0, entries in `/etc/exports` for DMS clients looked like this:

```
/dlclient0 -n -r=0 nobody
/dlclient0/clientA.root clientA
/dlclient0/clientB.root clientB
```

In ULTRIX/UWS Version 4.0 and ULTRIX/UWS Version 4.1, the first line is no longer necessary, since the `-r` rootmap option is now specified at the exported client root level. In addition, the `-n` option for "no filehandle" is no longer applicable. As a result, in ULTRIX/UWS Version 4.0 and ULTRIX/UWS Version 4.1, the `/etc/exports` entries in the preceding example should be changed to look like this:

```
/dlclient0/clientA.root -r=0 clientA
/dlclient0/clientB.root -r=0 clientB
```

If a client swaps over the network to the server, the `/etc/exports` entry for the swap file must be updated to ensure that the client has access to the swap file. If the client does not have the correct swap file when booted, it will crash.

To prevent this problem, add an entry for the client's swap in the server's `/etc/export` file. For example, if clientA is swapping over the network, edit the following entry:

```
/dlclient0/clientA.root -r=0 clientA
```

to explicitly add an entry for network swap, as follows:

```
/dlclient0/clientA.root -r=0 clientA
/dlclient0/clientA.root/dev/swap -r=0 clientA
```

1.5.5.2 The makpkt File Location Has Changed – When executing, DMS will run `makpkt`. The `makpkt` file used to be under `/usr/diskless`. In ULTRIX/UWS Version 4.0, the file was moved to `/usr/etc`. If the diskless server is upgraded to ULTRIX/UWS Version 4.1 and the diskless client area is ULTRIX Worksystem Software (UWS) Version 2.1 and UWS Version 2.2, the diskless server will have problems running the diskless client area since the `makpkt` file has been moved.

To work around the problem, make a symbolic link in the server system from `/usr/etc/makpkt` to `/usr/diskless/makpkt` by typing the following command:

```
# ln -s /usr/etc/makpkt /usr/diskless/makpkt
```

1.5.5.3 DMS Clients and Time Zone Information – After adding a diskless client to a diskless server and booting the diskless client, the client will build its own kernel. The kernel build process needs a timezone entry in the system configuration file of DMS clients.

The time zone information is obtained from the DMS Server's configuration file. The location of system configuration files changed in ULTRIX/UWS Version 4.0, causing DMS to fail to get the timezone entry. The system configuration file that used to be located under `/sys/conf` is now located under `/sys/conf/vax` (for VAX machines) or `/sys/conf/mips` (for RISC machines).

To work around the problem on the server system, make a symbolic link from `/sys/conf/vax/`*HOSTNAME* or `/sys/conf/mips/`*HOSTNAME* to `/sys/conf/`*HOSTNAME*, replacing the italic *HOSTNAME* with the name of the server in capital letters.

On a RISC server, use a command similar to the following:

```
# ln -s /sys/conf/mips/HOSTNAME /sys/conf/HOSTNAME
```

On a VAX server, use a command similar to the following:

```
# ln -s /sys/conf/vax/HOSTNAME /sys/conf/HOSTNAME
```

1.5.5.4 Different Versions of ULTRIX, DMS, and BIND – In ULTRIX-32 Version 3.1, if the diskless server is running BIND, diskless clients are automatically set up with BIND. In ULTRIX/UWS Version 4.1, if a diskless client desires the BIND service, `bindsetup` must be run from the diskless client, not from the server.

If the diskless server is upgraded to ULTRIX/UWS Version 4.1 and the existing ULTRIX Worksystem Software Version 2.1 and Version 2.2 diskless clients have already been set up with the BIND service, the diskless server will have problems manipulating those diskless client areas since its hostname does not include the BIND domain name.

For an ULTRIX/UWS Version 4.1 diskless server running BIND with existing ULTRIX Worksystem Software Version 2.1 and Version 2.2 diskless clients already set up as BIND clients, set the server's hostname to its name plus the BIND domain name in `/etc/rc.local` to ensure proper diskless client area manipulation by the diskless server.

1.6 Remote Installation Services (RIS)

This section contains notes about the Remote Installation Services (RIS).

1.6.1 RIS Clients

Remote Installation Services clients should only be registered with one RIS server at a time. Registering with more than one server causes unpredictable results.

1.6.2 Client Name Length

The first six characters (letters or numbers) of the name of either RIS or DMS clients registered to be served from an ULTRIX server must be unique. The server's registration facilities truncate all names to the first six characters.

For example, the names system1 and system2 are both registered as system, with the registration information of the second client replacing the registration information of the first.

1.6.3 Previous Versions of ULTRIX

The ULTRIX/UWS Version 4.1 RIS services do not support installations between clients and servers from different operating system versions. That is, you cannot install ULTRIX-32 Version 3.0 clients from an ULTRIX/UWS Version 4.1 server and you cannot install ULTRIX/UWS Version 4.1 clients from an ULTRIX-32 Version 3.0 server.

Previously, you could remotely install clients and servers from different versions of the ULTRIX operating system.

1.6.4 CDROM Usage in the RIS Environment

When a CDROM is used for a RIS environment, its throughput degrades as the number of concurrent RIS system loads increases. The decrease is additive, based on the number of active clients. For example, it takes about 8 minutes for the BASE subset to load for one VAX RIS client and about 17 minutes for the BASE subset to load for two VAX RIS clients.

1.6.5 Optional Removal of the Kernel Object Subset

ULTRIX/UWS Version 4.1 contains a kernel object subset which allows full debug capabilities using the `dbx` debugger.

In order to save space, you may remove your kernel object subset from your installed system. This can be accomplished using the following `setld` command:

```
# /etc/setld -d UDTBIN410
```

It is important that you understand some of the trade-offs you make by removing this subset.

If and when you need to apply a kernel patch to your environment, you will need to reinstall the kernel object subset. This might require that you remove other system or user environments to make room for its installation. (This assumes that the space made available after removing the subset was used for another purpose.)

In general, any time you need to build a kernel for your system, the subset will need to be reinstalled. This could be for a simple need to modify some hardware or software configuration parameter in the system's configuration file.

We recommend that this procedure not be followed routinely. Rather, it is intended to solve space problems on smaller systems with smaller system disks. If you choose to remove the kernel object subset, wait until the system is properly configured and proven for some reasonable period of time.

1.6.6 Installation of RIS Clients

There is a problem with installing RIS clients when the RIS Client Database is too large. The symptoms of the problem are messages displayed on the console of the client that indicate that the `root` file system is full. The messages typically are:

```
/: file system full
/: write failed, file system is full ?
```

The first time that these messages are displayed, the installation usually finishes without any problems. However, if you have registered a large number of RIS clients at once, the first time you see the error message could be fatal. You can have approximately 150 RIS clients registered before this problem occurs.

To work around this problem, deregister RIS clients that are no longer actively installing software from the RIS server.

To determine if the problem is about to happen, check the size of the RIS Client Database (`~ris/clients/risdb`) on the server. If this database is greater than 9 Kbytes in size, this problem could occur.

1.6.7 Mounting a RIS Area Using NFS

You can use an NFS mount point to set up a new RIS area by using a RIS area that exists on another machine. For example, if the server `tigris` has a CDROM optical disc containing ULTRIX RISC-supported subsets mounted on `/mnt`, the system manager on `jaguar` could type the following command to NFS-mount those subsets:

```
# mount tigris:/mnt/RISC/BASE /mnt
```

Then, the system manager on `jaguar` would invoke the `/etc/ris` utility and proceed with the procedure for installing software described in the *Guide to Remote Installation Services*.

1.6.8 Extracting Software or Creating a Symbolic Link from a CDROM

When you install software to a RIS area using CDROM media, you can either extract the software from the mount point or to create a symbolic link to the mount point. For example, if you set up an environment to serve RISC clients, a message like the following appears:

```
Choose one of the following options:

1  Extract software from /mnt/RISC/BASE
2  Create symbolic link to /mnt/RISC/BASE

Enter your choice:
```

You must enter a 1 or a 2 to indicate your choice.

Processor-Specific Notes 2

This chapter contains ULTRIX and ULTRIX Worksystem Software release notes specific to the following processors:

- MicroVAX II, VAXstation II, and VAXstation II/GPX
- VAXstation 2000 and MicroVAX 2000
- VAXstation 3520 and VAXstation 3540
- VAX 11/780 and 11/785
- VAX 6000 Model 400 and Model 500 Series
- VAX 8700 VAX 8800 Systems
- DECstation/DECsystem 2100 and 3100
- DECstation/DECsystem 5000 Model 200 Series
- DECsystem 5100
- DECsystem 5400
- DECsystem 5500
- DECsystem 5800 Series

2.1 MicroVAX II, VAXstation II, and VAXstation II/GPX

These notes apply to the MicroVAX II, VAXstation II, and VAXstation II/GPX.

2.1.1 Disabling Bootable Disks on MicroVAX II, VAXstation II, and VAXstation II/GPX Systems

The boot programs residing in PROM search for a bootable disk using a specific priority scheme. Removable disks are searched first, followed by nonremovable disks. For example, if your system resides on disk unit 1 but disk unit 0 always boots after a power failure or as a result of typing BOOT at the console prompt (>>>), you can disable disk unit 0's boot block.

To disable a particular disk unit's boot block, log in as `root` and type the following command:

```
# dd if=/.profile of=/dev/rraNa count=1
```

Be sure to select the drive you want to disable by replacing the italic *N* in the preceding example with the actual RA number of the disk you are disabling. Once you have disabled the disk, the boot program will skip the disabled disk in its search for a bootable disk.

2.1.2 Console Port Printer Procedure: MicroVAX Systems

The following procedure explains how to attach a console port printer to a MicroVAX system in a BA23 or BA123 enclosure. This procedure applies to MicroVAX systems that do not have a multiplexer at the time that the ULTRIX operating system is installed.

To connect the printer after installing ULTRIX software, follow these steps:

1. Open the back of the enclosure, if applicable.
2. Set the HALT ENABLE/DISABLE switch on the back of the system to the DISABLE position.
3. Set the console select switch to the proper speed for your printer.
4. Log in as `root` or become superuser.
5. Type these commands:

```
# cd /dev
# MAKEDEV ttycp
# ln ttycp lp
# chown daemon lp
# chmod 660 lp
```

6. Use the `lprsetup` utility or manual printer setup procedure as described in the *Guide to System Environment Setup*.

2.1.3 VAX Color Workstations

The following notes apply to VAX color workstations.

2.1.3.1 The Xqdsg Server – The `Xqdsg` server:

- Allows you to specify a plane mask in the gc.
- Supports five types of visuals for the root window. The default can be changed by using a command line option in `/etc/ttys`:

```
-class <classname>   type of Visual for root window,
  one of StaticGray, StaticColor, PseudoColor,
  GrayScale, or even TrueColor
```

- Provides performance improvements for all filled rectangles (stippled, tiled, and solid fill) and dashed lines.

2.1.3.2 Console Messages on VAX Color Displays – When the X server is running and a console window is not provided, system messages that are sent to the console on 8-plane systems are displayed as blank black lines beginning at the left edge of the screen. In addition, the `XPrompter` dialog box or the other portions of the display might be corrupted. Press the `Clear` button in the `Xprompter` dialog box to remove the corruption.

2.2 VAXstation 2000 and MicroVAX 2000

These notes apply to the VAXstation 2000 and the MicroVAX 2000.

2.2.1 Special File Usage: VAXstation 2000 and MicroVAX 2000

On VAXstation 2000 and MicroVAX 2000 processors, do not create or attempt to use the `/dev/tty00` special file, because it interferes with the operation of the console device.

For the VAXstation 2000, do not attempt to use the `/dev/tty01` special file because it interferes with the operation of the mouse. You can use this file on the MicroVAX 2000.

Refer to `ss`(4) in the *ULTRIX Reference Pages* for information on how the `/dev/tty??` files map to the four ports on the basic serial line unit (SLU).

2.2.2 Changing Speed on the Console Device

On the VAXstation 2000 and MicroVAX 2000, `stty` cannot change the speed on the console device. The console subsystem firmware requires the console terminal to operate at a fixed speed of 9600 bits per second (bits/s) for a CRT or hardcopy terminal, or 4800 bits/s for a graphics display device, such as a VR260 monitor. The console device must be set for 8-bit character length with one stop bit and no parity. The `ss` driver enforces these restrictions by disallowing some functions of the `stty` command, such as changing the line speed on the console port. For more information, see `ss`(4) in the *ULTRIX Reference Pages*. These restrictions apply only to the console device.

2.2.3 VAX Color Workstations

The following notes apply to VAX color workstations.

2.2.3.1 The Xqdsg Server – The `Xqdsg` server:

- Allows you to specify a plane mask in the gc.
- Supports five types of visuals for the root window. The default can be changed by using a command line option in `/etc/ttys`:

```
-class <classname>   type of Visual for root window,
  one of StaticGray, StaticColor, PseudoColor,
  GrayScale, or even TrueColor
```

- Provides performance improvements for all filled rectangles (stippled, tiled, and solid fill) and dashed lines.

2.2.3.2 Console Messages on VAX Color Displays – When the X server is running and a console window is not provided, system messages that are sent to the console on 8-plane systems are displayed as blank black lines beginning at the left edge of the screen. In addition, the `XPrompter` dialog box or the other portions of the display might be corrupted. Press the `Clear` button in the `Xprompter` dialog box to remove the corruption.

2.3 VAXstation 3520 and VAXstation 3540

These notes apply to the VAXstation 3520 and VAXstation 3540.

2.3.1 Advanced Installations Required for VAXstation 3520 and 3540 Systems

Advanced installations are required for VAXstation 3520 and 3540 systems. When selecting optional software subsets, you must choose the subset described as "VS35XX X11/DECwindows Fonts".

2.3.2 The X Server and Clients

The following notes apply to the X server:

- During the initialization of the X server, a cursor block may appear in the middle of the screen. As soon as the X server has started, this cursor disappears.
- The `ico` applications from MIT do not generate a faceted display on a 24-plane VAXstation 3520/3540. This is a problem within the `ico` application, as the `ico` application assumes the default colormap is writable.
- If you draw wide dash lines one pixel long with projecting caps, the server may crash.
- Three-dimensional perspective projections from inside objects are not clipped properly.
- On the VAXstation 3520/3540 processors, some incompatibilities exist between shared-memory transport and 3-dimensional graphic applications. Shared-memory should not be used if you are running 3-dimensional graphic applications (`local:0` and `:0`). If you are running 3-dimensional graphics locally, the best performance occurs when you use `iphost:0` over the TCP/IP network or `dnhost::0` when running over the DECnet network.
- The print screen option of the session manager does not work on 24-plane systems. It appears to work but, upon completion, does not create the output file.
- The color intensity labels on the VAXstation 3520/3540 Customize Window selections of the Session Manager may be clipped.
- The Red Intensity, Green Intensity, and Blue Intensity labels for the color adjustment of the Window/Screen foreground/background/highlight/border colors are clipped so that only the tops of the letters are visible. To correct the problem edit the `/usr/lib/X11/app-defaults/SessionManager` file and remove the 3 lines listing the *Color Attributes.*Scale.height resources.

2.3.3 The Xgb Server

The `Xgb` server uses a different font set, font compiler, color database, and PostScript Previewer than the other servers in this release. To avoid confusion, these components have been renamed for use with the `Xgb` server. These names are resolved during installation, but when running the font compiler or the Previewer you need to know the new names, which are listed in the following table:

Component	UWS Versions 2.0/2.1	UWS Versions 2.2/4.0/4.1
Fonts	`/usr/lib/dwf`	`/usr/lib/dwf`
Font compiler	`/usr/bin/dxfc`	`/usr/bin/dxfc3d`
Color database	`/usr/lib/X11/rgb.*`	`/usr/lib/rgb.*`
PostScript previewer	`/usr/bin/dxpsview`	`/usr/bin/dxpsview3d`

2.3.4 VAX Color Workstations

The following notes apply to VAX color workstations.

2.3.4.1 The Xqdsg Server – The `Xqdsg` server:

- Allows you to specify a plane mask in the gc.
- Supports five types of visuals for the root window. The default can be changed by using a command line option in `/etc/ttys`:

```
-class <classname>   type of Visual for root window,
  one of StaticGray, StaticColor, PseudoColor,
  GrayScale, or even TrueColor
```

- Provides performance improvements for all filled rectangles (stippled, tiled, and solid fill) and dashed lines.

2.3.4.2 Console Messages on VAX Color Displays – When the X server is running and a console window is not provided, system messages that are sent to the console on 8-plane systems are displayed as blank black lines beginning at the left edge of the screen. In addition, the `XPrompter` dialog box or the other portions of the display might be corrupted. Press the `Clear` button in the `Xprompter` dialog box to remove the corruption.

2.4 VAX 11/780 and 11/785

This note applies to the VAX 11/780 and 11/785.

2.4.1 Boot Command for the VAX 11/780 and 11/785

The *Basic Installation Guide* omits the console mode bootstrap command in its instructions for the VAX-11/780 or VAX-11/785 processors in Section 2.11. The procedure to boot the system should include the boot command, as shown here:

```
>>> b
```

2.5 VAX 6000 Model 400 and Model 500 Series Processors

The following notes apply to the VAX 6000 Model 400 and Model 500 Series processors.

2.5.1 Installation Instructions for the VAX 6000 Model 500

The installation instructions for the VAX 6000 Model 400 in the *Basic Installation Guide* and the *Advanced Installation Guide* support installations of the VAX 6000 Model 500.

2.5.2 Missing Boot Commands

When the installation completes on a VAX 6000 Model 400 and Model 500 Series processor, the system does not display instructions for updating the console boot defaults. The following procedure explains how to update the missing console boot defaults:

1. During the installation, immediately after the root partition is restored, the system displays a message like the following:

```
******BOOTSTRAP COMMAND SEQUENCE******

Enter the following boot sequence at the console mode prompt
after the installation halts the processor:

Make sure that the console TK50 tape is in the drive before entering
the boot command.

>>> b /xmi:e /bi:4 du9

.
.
.
```

   Record the boot parameters. You will need them later on. The boot parameters in the preceding example are `/xmi:e`, `/bi:4`, and `du9`.

2. When the processor halts, instead of typing the bootstrap command sequence, turn the key switch on the front panel of your processor to the position labeled "Update".

3. At the console prompt, enter commands with the following syntax to set your boot defaults:

   set boot default /r5:10008 *boot_param1 boot_param2 boot_param3*
   set boot ask /r5:1000b *boot_param1 boot_param2 boot_param3*

   For example, using the boot parameters in the preceding example, you would type the following commands:

```
>>> set boot default /r5:10008 /xmi:e /bi:4 du9
>>> set boot ask /r5:1000b /xmi:e /bi:4 du9
```

4. Set the key switch to the "Enable" position.

5. Now when the installation completes, although the instructions for updating the console boot commands fail to appear, you can boot the default system device to multi-user mode by typing the following command:

```
>>> b
```

You can boot the default system device to single-user mode by typing the following command:

```
>>> b ask
```

2.5.3 Installation Problem

During the installation procedure for VAX 6000 series processors, the installation procedure displays the disk unit number in the boot command as a hexadecimal number. If you enter the hexadecimal number as the disk unit number, the boot command fails.

To work around the problem, convert the disk unit number supplied by the installation procedure to a decimal number and use that decimal number in the boot command.

2.6 VAX 8700 and VAX 8800 Systems

This note applies to the VAX 8700 and VAX 8800 Systems.

2.6.1 Maximum Memory Support for VAX 8700 VAX 8800 Systems

A maximum of 448 Mbytes of memory is currently supported on the VAX 8700, VAX 8800, VAX 8820, VAX 8830, and VAX 8840 systems. Any of these systems configured with 512 Mbytes of memory will not install ULTRIX/UWS Version 4.1.

If you require full use of the 512 Mbytes of memory for these processors, please contact your Customer Support Center for assistance.

2.7 DECstation/DECsystem 2100 and 3100

These notes apply to the DECstation/DECsystem 2100 and DECstation/DECsystem 3100.

2.7.1 Getting a Memory Dump from a Hung DECstation/DECsystem 3100

If a DECstation/DECsystem 3100 "hangs", you can press the reset button to enter console mode. The default action on the DECstation/DECsystem 3100 is for the reset to reinitialize memory. To prevent this (preserve memory), set the bootmode to debug by typing the following command in console mode:

```
>>> setenv bootmode d
```

Then, should a "hang" occur, you can press reset to return to console mode (with contents preserved) and obtain a memory dump. The memory dump routine can be run by typing the go command with the address:

```
>>> go 0x80030008
```

If the system is in multi-user mode when the reset button is pressed, then the dump will occur silently and no messages will be printed. The memory dump will take several minutes, then the console prompt will reappear. After the dump is completed, you can reinitialize the system and reboot as follows:

```
>>> init
>>> auto
```

Note

When bootmode is set to `d` it is important to type **init** before typing `boot` or `auto` when the system has been shutdown to console mode or reset to console mode. Failure to use the `init` command may cause the system boot to fail.

2.7.2 Terminal Emulator Windows

There is a problem on the DECstation/DECsystem 2100 and the DECstation/DECsystem 3100 that manifests itself when the login shell is `/bin/sh` and the user is `root`. It may take as long as 3 minutes before terminal emulator windows appear.

You can work around this problem as follows:

1. Edit the file `/.profile`.
2. Move the line beginning with "echo ..." and reinsert it directly after the line beginning with "stty ..."

The file should now appear similar to the following:

```
#       @(#).profile    4.3     ULTRIX  11/18/88
stty dec crt new
echo 'erase ^?, kill ^U, intr ^C'
umask 22
PATH=/usr/ucb:/bin:/usr/bin:/etc:/usr/local:/usr/new:/usr/hosts:.
export PATH
```

2.7.3 Xcfb Server

Under the `Xcfb` server, endpoints of zero-width capped polylines may be drawn in the wrong location if the window is partially clipped. When this bug manifests itself, apparently random singleton points will be seen. The workaround is to ensure that the window is not clipped (for example, by raising the window to the top of the stack).

2.7.4 Invoking dxmail from the User Executive

Using `dxmail` invoked from the User Executive Option disables sending or receiving mail. If you run the `csh` shell, you can work around this problem by invoking `dxmail` from the command line after putting `/usr/new/mh` in your `PATH`. This solution will not work when using the Bourne shell.

2.8 DECstation/DECsystem 5000 Model 200 Series

The following notes apply to the DECstation/DECsystem 5000 Model 200 series.

2.8.1 Booting the DECstation/DECsystem 5000 Model 200 Series Processor

Due to changes in the firmware of the DECstation/DECsystem 5000 Model 200 series processors to enhance support for the TURBOchannel, the boot commands have changed. As a result, the boot commands for the DECstation/DECsystem 5000

Model 200 series processors that are listed in the *Basic Installation Guide, Guide to System Shutdown and Startup, Guide to Diskless Management Services,* and that are displayed on the console during installation are no longer correct.

After your new firmware is installed, even if you are not reinstalling the ULTRIX operating system, you must immediately reset the environmental variables for the `boot` and `haltaction` commands.

Until the software and documentation can be revised, please refer to this document when booting your DECstation/DECsystem 5000 Model 200 series processor.

The following sections explain the procedures for:

- Determining the slot and unit numbers of your boot device
- Setting the console environmental variables
- Booting from a system disk
- Booting from a TK50 tape
- Booting from a CDROM disk kit
- Booting from the network
- Booting during the installation

2.8.1.1 Determining the Slot and Device Numbers of Your Boot Device – If you are not reinstalling the ULTRIX operating system after your new firmware has been installed, you will need to determine both the slot number of the controller attached to your boot device and the device number of your boot device in order to change the `boot` variable.

The instructions in this section assume that if you have multiple disk drives, CDROM drives, or tape drives, you know which drive is your boot device.

The instructions in this section also assume that you are booting your system from a logical device controller with a controller number of 0 (0 is the default boot controller number).

This is the most common configuration.

If you are booting from other than controller 0, check the cabling of your hardware to determine which controller you are booting from or, if you are up and running, check the SYSTEM DISK SELECTION section in the `install.log` file in the `/usr/var/adm` directory. Then refer to Section 2.8.1.7.1, Installing as a Standalone, or Section 2.8.1.7.2, Installing as a Diskless Client, of this document for instructions on how to determine the slot number of the controller attached to your boot device.

2.8.1.1.1 Determining the Slot Number (Default) – To determine which controllers and devices are configured on your system, type the following command at the console prompt:

```
>> cnfg
```

The `cnfg` command displays the options present on the system, as follows:

```
7:KN0Z-AA     DEC    V5.3a      TCF0       (16 MB)
6:PMAD-AA     DEC    V5.3a      TCF0       (enet:08)
5:PMAZ-AA     DEC    v5.3a      TCF0       (SCSI=7)
```

```
2:PMAZ-AA     DEC    v5.3a     TCF0      (SCSI=7)
1:PMAG-AA     DEC    T3.0a     TCF0      (PXG--D=24)
```

The first column always displays slot numbers of the device controllers; the slot numbers differ with each configuration.

However, when you boot your system from a logical device controller with a controller number of 0 (the default boot controller), the default slot number for the default SCSI boot device is always 5 and the default slot number for the default network boot device is always 6.

Table 2-1 lists the boot information for default SCSI devices that are attached to the controllers identified by the letters "SCSI" in the fifth column of the `cnfg` command display. Table 2-2 lists the boot information for default network devices that are attached to the controllers identified by the letters "enet" in the fifth column of the `cnfg` command display.

Table 2-1: Default SCSI Devices

Boot Devices	`cnfg` Controller Identifiers	Device Name	Default Slot Number	Default Controller Number
Tape	SCSI	tz	5	0
Disk	SCSI	rz	5	0
CDROM	SCSI	rz	5	0

Table 2-2: Default Network Devices

Boot Devices	`cnfg` Controller Identifiers	Device Name	Default Slot Number	Default Controller Number
Network Options	enet	–-	6	0

2.8.1.1.2 Determining the Boot Device Number – To determine the device number of your boot device, type a command with the following syntax:

cnfg *slot_number*

For example, to determine the device number of a SCSI disk drive with a controller number of 0 and a slot number of 5, type the following command:

```
>> cnfg 5
```

The disk drive is identified by the letters "rz" in the command's display and the device number immediately follows the "rz" (rz3, in the following example):

```
5:     PMAZ-AA   DEC     V5.3a     TCF0          (SCSI = 7)
-------------------------------------------------------------

          DEV    PID               VID      REV   SCSI DEV
-------------------------------------------------------------
-------------------------------------------------------------
          rz3    RZ56   (C)DEC     DEC      0200    DIR
                                                    SEQ
```

You are then able to boot your system or set your system's environmental boot variable, since you know that the slot number of controller 0 is 5, the device name is rz, and device number of the boot device attached to that controller is 3 (rz3).

Note

If the `cnfg` *slot_number* command shows that you have multiple devices configured on the same controller, you will have to know which device is the boot device. If you do not know which device is the boot device, consult your system administrator.

2.8.1.2 Setting the Console Environmental Variables – The following sections explain how to set the console environmental variables for the `boot` and `haltaction` commands.

After your new firmware is installed, even if you are not reinstalling the ULTRIX operating system, you must immediately reset the environmental variables for the `boot` and `haltaction` commands. You can also set other console environmental variables. To get a listing of the variables, type:

```
>> printenv
```

For more information about the variables and for instructions on how to set each, see your hardware manual.

2.8.1.2.1 Setting the boot Variable – You can define the default bootpath and enable or disable automatic boot operations by setting specific console environmental variables, depending on whether you will be booting from the system disk or from the network.

Setting the System Disk Boot Variable

The `boot` variable sets the default boot device. To set the `boot` variable for the system disk, use a command with the following syntax:

setenv boot *slot_number/device_name_number/unix_kernel* [-a]

Replace *slot_number* with the slot number of the disk controller that is to be the default boot device. Replace *device_name_number* with the name and the device number of the default boot device. Replace *unix_kernel* with the pathname of the UNIX kernel that is to be the default kernel.

Use the `-a` switch to enable booting to multi-user mode by default. Please note that if you use the `-a` switch, everything after the word "boot" must be surrounded by double quotation marks (").

Multiuser Mode – To set the `boot` environmental variable to boot to multi-user mode by default, you must set the `-a` switch and surround everything after the word "boot" in double quotation marks (").

For example, to set the default boot device to an rz disk at slot 0, drive 1, with `vmunix` as the default kernel, and to set the default to boot to multi-user mode, type the following command:

```
>>  setenv boot "0/rz1/vmunix -a"
```

Single-user Mode – To set the `boot` environmental variable to boot to single-user mode by default, do not set the `-a` switch.

For example, to set the default boot device to an rz disk at slot 0, drive 1, with `vmunix` as the default kernel, and to set the default to boot to single-user mode, type the following command:

```
>> setenv boot 0/rz1/vmunix
```

Setting the Network Boot Variable

The `boot` variable sets the default boot device. To set the `boot` variable for the network, use a command with the following syntax:

setenv boot *slot_number*/mop [-a]

Replace *slot_number* with the slot number of the module that is to be the default boot device.

Use the `-a` switch to enable booting to multi-user mode by default. Please note that if you use the `-a` switch, everything after the word "boot" must be surrounded by double quotation marks (").

Multiuser Mode – To set the `boot` environmental variable to boot to multi-user mode by default, you must set the `-a` switch and surround everything after the word "boot" in double quotation marks (").

For example, to set the default boot device to the network at slot 0, and to set the default to boot to multi-user mode, you would type the following command:

```
>>  setenv boot "0/mop -a"
```

Single-user Mode – To set the `boot` environmental variable to boot to single-user mode by default, do not set the `-a` switch. For example, to set the default boot device to the network at slot 0, and to set the default to boot to single-user mode, type the following command:

```
>> setenv boot 0/mop
```

2.8.1.2.2 Setting the haltaction Variable – The `haltaction` variable enables or disables automatic boot operation. To set the `haltaction` variable, use a command with the following syntax:

setenv haltaction *variable*

To enable automatic boot mode using the boot variable, set the `haltaction` variable to `b` by typing the following command:

```
>> setenv haltaction b
```

Note

If you wish to enable automatic rebooting to multi-user mode, you must make sure that when you initially set the boot variable, you use the `-a` switch.

To disable the automatic boot operation (that is, to suppress an automatic reboot after the RESET button has been depressed or as the result of a power on), set the `haltaction` variable to `h` by typing the following command:

```
>> setenv haltaction h
```

To force the system to restart when the reset button is pressed, and thereby do a memory dump, set the `haltaction` variable to `r` by typing the following command:

```
>> setenv haltaction r
```

2.8.1.3 Booting from a Disk – You can boot the system disk or an alternate disk or alternate kernel to either single-user or multi-user mode.

2.8.1.3.1 Booting from the System Disk – To boot the system disk to single-user or multi-user mode, type the following command:

```
>> boot
```

The system boots the device that was set in the `boot` console environmental variable described previously.

Note

If you wish to boot the default disk or kernel image to multi-user mode, you must make sure that when you initially set the boot variable, you use the `-a` switch.

2.8.1.3.2 Booting from an Alternate Disk or Kernel – To boot an alternate disk or kernel image to single-user or multi-user mode use a command with the following syntax:

boot *slot_number/device_name_number/unix_kernel* [-a]

Replace *slot_number* with the slot number of the controller attached to your boot device. Replace *device_name_number* with the name and the number of the boot device. Replace *unix_kernel* with the pathname of the alternate kernel. Use the `-a` switch to enable booting to multi-user mode.

For example, to boot an alternate kernel at slot 0, drive 5 to multi-user mode, type the following command:

```
>> boot 0/rz5/vmunix.new -a
```

2.8.1.4 Booting from a TK50 Tape – When doing an installation or, when booting the standalone kernel for system management tasks, you may have to boot a TK50 tape. After installing the TK50 boot tape, type a command with the following syntax to determine the device number of the drive for your device:

cnfg *slot_number*

For example, if the TK50 is attached to a SCSI controller at slot 5 (this is the most common configuration), type the following command:

```
>> cnfg 5
```

The console subsystem displays information that identifies the device number of your tape drive and various other assignments. Use this information to define the tape drive device number when you enter the boot command later.

After displaying identification information, the console subsystem reissues its prompt. Use a command with the following syntax to boot your system:

boot *slot_number*/tz*device_number*

Replace *slot_number* with the slot number of the tape controller. Replace *device_number* with the device number of the SCSI tape drive from which you are booting.

For example, to boot a SCSI tape (tz) at slot 5, drive 5 to single-user mode, type the following command:

```
>> boot 5/tz5
```

2.8.1.5 Booting from a CDROM Optical Disk Kit – If your CDROM optical disk is not already in its caddy, follow the instructions in the hardware manual for inserting the optical disk into the caddy.

To boot the system, load the CDROM optical disk into the drive and wait for the drive to be on line and ready.

Use a command with the following syntax to determine the device number of the drive for your device:

cnfg *slot_number*

For example, if the CDROM is attached to a SCSI controller at slot 5 (this is the most common configuration), type the following command:

```
>> cnfg 5
```

A display appears that shows what is assigned to each device number on your system. Use a command with the following syntax to boot your system:

boot *slot_number*/rz*device_number*/vmunix [-a]

Replace *slot-number* with the slot number of the CDROM controller. Replace *device-number* with the device number of your RRD40 optical disk drive. Use the `-a` switch to enable booting to multi-user mode.

For example, to boot the system to multi-user mode from RRD40 optical disk drive number 4 on slot number 1, type the following command:

```
>> boot 1/rz4/vmunix -a
```

2.8.1.6 Booting from the Network – You boot your system from the network when you are:

- Booting a diskless system
- Initiating an installation from a remote server
- Booting a standalone kernel from a remote server, in order to perform system management tasks

To boot the system from the network, use a command with the following syntax:

boot *slot_number*/mop [-a]

Replace *slot_number* with the slot number of the network controller. Use the **-a** switch to enable booting to multi-user mode.

For example, to boot from the network to multi-user mode on slot number 6, type the following command:

```
>> boot 6/mop -a
```

2.8.1.7 Booting During the Installation – This section explains how to boot your system from the system disk or, if your system is a diskless client, from the kernel residing on the diskless server during the installation of your ULTRIX operating system.

During the installation, the bootstrap command sequence is displayed. You are asked to type this command sequence in order to boot your system from the system disk to continue the installation or, if your system is a diskless client, from the kernel residing on the diskless server.

Because of changes made to the firmware to enhance support for the TURBOchannel, both the syntax of the bootstrap command sequence and the slot number that are displayed are incorrect.

The following instructions explain how to determine the correct slot number for your system configuration so that you can type the correct bootstrap command sequence and continue the installation.

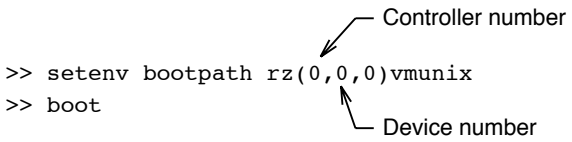
2.8.1.7.1 Installing as a Standalone Machine – During the installation, after the system loads the kernel image into main memory and finishes configuring, a bootstrap command sequence like the following is displayed:

```
*** BOOTSTRAP COMMAND SEQUENCE ***

Issue the following console commands to set your default bootpath
variable and to boot your system disk:

        >> setenv bootpath rz(0,0,0)vmunix
        >> boot


The system name assigned to your processor is calypso.
.
.
.
```

This bootstrap command sequence is incorrect. To determine the correct bootstrap command sequence, you need to note the controller number and the device number that are displayed. The controller number enables you to determine the correct slot number of the controller that is attached to your disk. You will use the device

number, exactly as it is displayed, to reboot your system. The controller number and the device number used in the preceding example are identified for you in Figure 2-1.

Figure 2-1: Bootstrap Command Sequence: Standalone



Write down both the controller number and the device number displayed by the bootstrap command sequence. You will need them later.

Controller Number Is 0

If the bootstrap command sequence that displays on your system console contains a controller number of 0, then the correct slot number is 5. This is the most common configuration.

Type the following bootstrap command sequence to reboot your system when the incorrect bootstrap command sequence appears on your console:

```
>> setenv boot "5/rz0/vmunix -a"
>> boot
```

After you type the bootstrap command sequence, the system reboots and your installation continues. Refer to the section on the DECstation/DECsystem 5000 Model 200 series in the *Basic Installation Guide* and the *Advanced Installation Guide* to complete the installation.

Controller Number Is Greater Than 0

If the bootstrap command sequence that displays on your system console contains a controller number greater than 0, then you have more than one SCSI controller configured on your system and you have chosen to boot from a system disk connected to a SCSI controller other than the default SCSI controller.

To determine the slot number of the SCSI controller of the system disk, follow these steps:

1. When the system prompts you to type the bootstrap command sequence, type the following command at the console instead:

   ```
   >> cnfg
   ```

   The `cnfg` command displays the options present on the system, as follows:

   ```
   7:KN0Z-AA    DEC    V5.3a     TCF0      (16 MB)
   6:PMAD-AA    DEC    V5.3a     TCF0      (enet:08)
   5:PMAZ-AA    DEC    v5.3a     TCF0      (SCSI=7)
   2:PMAZ-AA    DEC    v5.3a     TCF0      (SCSI=7)
   1:PMAG-AA    DEC    T3.0a     TCF0      (PXG--D=24)
   ```

   The first column displays slot numbers. Note the slot numbers for SCSI options; SCSI options are identified by the letters "SCSI" in the fifth column.

In the preceding example, the SCSI controller slot numbers that correspond to the SCSI options are 5 and 2.

2. Since your system disk is on a SCSI controller with a controller number greater than 0 (0 is the default controller number for the boot device), you can disregard slot number 5. Slot number 5 always corresponds to the default SCSI controller.

   Of all the SCSI controllers that are configured on your system, the lowest corresponding SCSI slot number displayed by the `cnfg` command, excluding slot number 5, corresponds to SCSI controller 1. The next highest SCSI slot number corresponds to SCSI controller number 2, and so forth.

   In the preceding example, the system disk is attached to SCSI controller 1 at slot number 2. The device number, identified in Figure 2-1, is 0.

3. You would therefore type the following bootstrap command sequence to reboot your system during the installation:

   ```
   >> setenv boot "2/rz0/vmunix -a"
   >> boot
   ```

   The syntax for this command sequence is as follows:

   setenv boot "*slot_number*/rz*device_number*/vmunix -a"
   boot

After you type the correct bootstrap command sequence, the system reboots and your installation continues. Refer to the section on the DECstation/DECsystem 5000 Model 200 series in the *Basic Installation Guide* and the *Advanced Installation Guide* to complete the installation.

2.8.1.7.2 Installing as a Diskless Client – During the installation, after the system loads the kernel image into main memory and finishes configuring, a bootstrap command sequence like the following is displayed:

```
*** BOOTSTRAP COMMAND SEQUENCE ***

After the system halts, type the following commands to set the
default bootpath to the network and reboot.

        >> setenv bootpath mop(0)
        >> boot


The system name assigned to your processor is saturn.
.
.
.
```

This bootstrap command sequence is incorrect. To determine the correct bootstrap command sequence, you need to note the controller number that is displayed; the controller number enables you to determine the correct slot number of the network controller you will use to reboot your system. The controller number in the preceding example is identified for you in Figure 2-2.

Figure 2-2: Bootstrap Command Sequence: Diskless


```
>> setenv bootpath mop(0)
>> boot
```

Controller Number Is 0

If the bootstrap command sequence that displays on your system console contains a controller number of 0, then the correct slot number is 6. This is the most common configuration.

Type the following bootstrap command sequence to reboot your system when the incorrect bootstrap command sequence appears on your console:

```
>> setenv boot "6/mop -a"
>> boot
```

After you type the bootstrap command sequence, the system reboots and your installation continues. Refer to the section on the DECstation/DECsystem 5000 Model 200 series in the *Basic Installation Guide* and the *Advanced Installation Guide* to complete the installation.

Controller Number Is Greater Than 0

If the bootstrap command sequence that displays on your system console contains a controller number greater than 0, then the diskless server has more than one network controller configured on its system and you are booting from a network controller other than the default network controller. To determine the slot number of the network controller from which you are booting follow these steps:

1. When the system prompts you to type the bootstrap command sequence, type the following command at the console instead:

   ```
   >> cnfg
   ```

   The `cnfg` command will display the options present on the system, as follows:

   ```
   7:KN0Z-AA     DEC    V5.3a      TCF0       (16 MB)
   6:PMAD-AA     DEC    V5.3a      TCF0       (enet:08)
   5:PMAZ-AA     DEC    v5.3a      TCF0       (SCSI=7)
   2:PMAZ-AA     DEC    v5.3a      TCF0       (SCSI=7)
   1:PMAD-AA     DEC    T3.0a      TCF0       (enet:08)
   ```

   The first column displays slot numbers. Note the slot numbers for network options; all network options are identified by the letters "enet" in the fifth column.

   In the preceding example, the slot numbers that correspond to the network controllers are 6 and 1.

2. Since you are to boot from a network controller that has a controller number greater than 0 (0 is the default controller number for boot devices), you can disregard slot number 6. Slot number 6 always corresponds to the default network controller.

Of all the network controllers that your diskless server has configured on its system, the lowest corresponding network slot number displayed by the `cnfg` command, excluding slot number 6, corresponds to network controller 1. The next highest network slot number corresponds to network controller number 2, and so forth.

In the preceding example, you are booting from network controller 1 at slot number 1.

3. You would therefore type the following bootstrap command sequence to reboot your system during the installation:

```
>> setenv boot "1/mop -a"
>> boot
```

The syntax for this command sequence is as follows:

setenv boot "*slot_number*/mop -a"
boot

After you type the correct bootstrap command sequence, the system reboots and your installation continues. Refer to the section on the DECstation/DECsystem 5000 Model 200 series in the *Basic Installation Guide* and the *Advanced Installation Guide* to complete the installation.

2.8.2 Installation Instructions for the Greyscale Monitor

When installing ULTRIX/UWS Version 4.1 on a DECstation/DECsystem 5000 Model 200PX with a color frame buffer display, you will be asked, during the server subset installation, if you are using a Greyscale Monitor. If your monitor is a VR262, answer yes (y). If not, answer no (n).

2.8.3 Interrupting the ULTRIX Operating System on a DECstation/DECsystem 5000 Model 200 Series Processor

If you set the `haltaction` variable to `r` (restart), then when you press the restart button, the system will dump core and reboot, instead of halting and clearing memory.

Note that the dump may be silent.

To set the `haltaction` variable to restart, enter the following command at the console:

```
>>> setenv haltaction r
```

If the system "hangs" or drops into console mode without doing a memory dump, you can start the memory dump routine manually.

If system "hangs", press the reset button to enter console mode. By default, the `haltaction` variable is set to reinitialize memory. To preserve memory, set the `haltaction` variable to debug mode by typing the following command at the console:

```
>>> setenv haltaction d
```

With `haltaction` set this way, if the system is "hung" you can press the reset button to enter console mode (with memory contents preserved). The crash dump code can then be run by typing the `go` command with a special address (the kernel start address + 8) that will call the memory dump routine, as follows:

```
>>> go 0x80030008
```

If the system is in multi-user mode when the reset button is pressed, the dump will occur silently and no messages will be printed. The memory dump will take several minutes, then the console prompt will reappear. After the dump is completed, you can reinitialize the system and reboot as follows:

```
>>> init
>>> boot
```

Note

When the `haltaction` variable is set to `d` it is important to type **init** before **boot** when the system has been shutdown to console mode, or reset to console mode. Failure to use the `init` command may cause the system boot to fail.

2.8.4 Custom Kernel Problem with DECstation/DECsystem 5000 Model 200 Series Processors

If you configure a DECstation/DECsystem 5000 to swap on boot, the machine will not boot because the asc driver was left out of the `genericconf` table in the file `machine/common/conf.c.` This problem does not affect the generic kernel (`genvmunix`), only custom kernels that are configured as follows:

```
config vmunix swap on boot
```

To avoid this problem, you can explicitly configure the root, swap, and dump devices. The following example shows the devices on unit number 21; you would substitute the unit number of your device:

```
config vmunix root on rz21a swap on rz21b dumps on rz21b
```

If you need to build a kernel for a configuration that must be bootable from multiple disks, you must first modify the file `machine/common/conf.c` as follows and then rebuild the kernel:

1. Copy the file `conf.c` to `conf.c.orig.`
2. Add an entry for the SCSI driver ascdriver after the entries for int scsidriver and int siidriver:

```
int scsidriver;
int siidriver;
int ascdriver;/* add this line */
```

3. Add the following lines after the existing entry for SII (shown below):

```
#if NSII > 0
extern struct uba_driver siidriver;
#endif

/* add the next three lines */
#if NASC > 0
extern struct uba_driver ascdriver;
#endif
```

4. Add an entry for the ASC for the BTD type after the entry for SII:

```
#define BTD$K_SII0
#define BTD$K_ASC0/* add this line */
```

5. Add an entry for the ASC driver at the end of the `genericconf` table:

```
#if NSII > 0
{ (caddr_t)&siidriver,"rz",makedev(21,0),BTD$K_SII },
#endif

/* add the next three lines */
#if NASC > 0
{ (caddr_t)&ascdriver,"rz",makedev(21,0),BTD$K_ASC },
#endif
```

6. Rebuild the kernel.

   For information on rebuilding your kernel, see Chapter 2 of the *Guide to Configuration File Maintenance*.

2.8.5 Writing, Adding, and Configuring a Device Driver for the TURBOchannel

The TURBOchannel is the system I/O bus on the DECstation/DECsystem 5000 Model 200 Series processor. This section discusses how to write and add a device driver for the TURBOchannel and provides information on the configuration of a user-written device driver for the TURBOchannel.

2.8.5.1 Writing a Device Driver – The device driver for the TURBOchannel has the same basic structure as a user-written device driver for a qbus or unibus device. You must be certain to provide the following information when creating a user-written driver:

- Define the `uba_driver` structure as shown in `/sys/io/uba/ubavar.h`. All user-written drivers on the TURBOchannel must define and use the `uba` data structures; however, the user-written driver does not have to access the `uba` mapping registers because the TURBOchannel is 32-bits wide.
- Include the TURBOchannel header file, `../io/tc/tc.h`, in addition to the header files normally included.
- Declare as **volatile** any variable or data structure that can be changed by a controller or processor other than the system CPU. Hence, variables that correspond to hardware device registers and any data structures or variable that is shared with a controller or coprocessor should be declared as **volatile**.
- Use the kernel routines. The callable kernel routines for the TURBOchannel include the following:

  tc_enable_option(ui) — This routine accepts as a parameter a pointer to a `uba_device` structure or a pointer to a `uba_ctlr` structure. This routine enables the device's interrupt line to the processor. Usually, a device driver does not use this routine; however, it must be used if the device has interrupts intermittently enabled and disabled during configuration or operation.

tc_disable_option(ui)	This routine accepts as a parameter a pointer to a `uba_device` structure or a pointer to a `uba_ctlr` structure. It disables the device's interrupt line to the processor. Usually, a device driver does not use this routine; however, it must be used if the device has interrupts intermittently enabled and disabled during configuration or operation.
bufflush(bp)	If the device driver performs DMA to host memory (as opposed to programmed I/O), the driver must explicitly flush the data cache, because a hardware cache coherency mechanism does not exist. The driver should call this routine after the DMA is complete, but before releasing the buffer (**bp**) to the system. The buffer pointer is specified by **bp**.
wbflush()	When the driver writes to a hardware device register, the write is delayed by the system write buffer. A subsequent read of that register does not wait for the write to complete. To ensure that a write to I/O space is complete, this routine should be called without arguments.

2.8.5.2 Adding a User-Written Driver – The following steps describe how to add a user-written driver for the TURBOchannel to the ULTRIX operating system:

1. Edit the file `/sys/data/tc_option_data.c` and create a new entry to the **tc_option** table. This table maps the device name in the ROM on the hardware device module to the driver in the ULTRIX kernel. The `/sys/data/tc_option_data.c` file contains instructions on where to place the new entry. Use the format of the default entries as a guide to format the new entry.
2. Add an entry to the system configuration file, `/sys/conf/mips/`YOURHOST. This entry must include the name of the device, the device unit number, and the name of the routine that handles the device's interrupts.
3. Add an entry for the driver file to the file `/sys/conf/mips/files.mips` as **Notbinary**. For example:

   ```
   io/tc/somedev.c    optional somedev device-driver Notbinary
   ```

4. Create an entry in the device configuration table `/sys/machine/mips/conf.c` to map the `/dev` special file entry to the driver routines in the kernel.
5. Compile and link the driver, then rebuild the kernel.

For information on how to rebuild the kernel, see the *Guide to Configuration File Maintenance*.

2.8.5.3 Configuration of the TURBOchannel User-Written Device Driver –

During system startup, the ULTRIX operating system probes the TURBOchannel option slots to determine which slots contain an IO module. Each option slot is at a fixed and known physical address; hence, the ULTRIX operating system can locate the option slots by their known physical address. Each IO module must have a ROM with a known format. To determine the name and the width (number of slots occupied) of the IO module, the ULTRIX operating system reads the ROM associated with that IO module.

Once the IO module name is determined, the ULTRIX operating system searches the **tc_option** data table to acquire the device or controller name as it appears in the system configuration file. The ULTRIX operating system compares the names of the devices (both optional and fixed) found in the IO slots with the device names specified in the configuration file (in the ub[md]init tables). Each configuration file entry specifies the interrupt routine name for the device. This information is placed in the ub[md]init tables during the configuration process.

For every device name that matches, an entry is placed in an internal table (**tc_slot**). These entries map the TURBOchannel slot numbers to device interrupt routines. The internal table contains a structure entry for each of the TURBOchannel IO options. All structure entries specify the name of the device, the slot number, the option width, the interrupt routine, unit number, and the base physical address of the device. This allows the operating system to call the correct interrupt handling routine for any slot that interrupts.

If the ULTRIX operating system encounters a module name in the module ROM that is not specified in the **tc_option** data table, the system displays a warning message which states the device is unknown. Known devices, such as SCSI and LANCE, are set up automatically in the **tc_option** data table. You may also include additional mappings for other module names to configuration file names.

The ULTRIX operating system calls the `probe`, `attach`, and `slave` routines through the *ibus* configuration routines for properly configured and recognized devices and controllers. The *ibus* configuration routines obtain the names of the `probe`, `attach`, and `slave` routines from the device drivers' `uba_driver` structure.

Adapters are handled similarly to devices and controllers. Each adapter has an adapter entry line in the system configuration file, but an interrupt routine name is not attached to it. The system autoconfiguration code searches for the adapter module name in the **tc_option** data table to obtain the name of the adapter configuration routine to call. In one of the arguments passed to the adapter configuration routine is an address where the adapter configuration routine then places the address of the interrupt handling routine.

The **tc_option** data table and the system configuration file provide a flexible mechanism for adding third party devices and device drivers. All device drivers must conform to the standard ULTRIX operating system conventions. For example, all drivers must have a `uba_driver struct` specifying the name of the device probe routine, the attach routine, the device name, and so on. The following example shows the structure for the Lance driver:

```
struct uba_driver lndriver =
  lnprobe, 0, lnattach, 0, lnstd, "ln", lninfo };
```

The corresponding entry in the system configuration file appears as follows:

```
device          ln0     at ibus?        vector lnintr
```

For more information on system configuration, see the *ULTRIX Guide to Configuration File Maintenance*.

2.8.6 Xcfb Server

Under the `Xcfb` server, endpoints of zero-width capped polylines may be drawn in the wrong location if the window is partially clipped. When this bug manifests itself, apparently random singleton points will be seen. The workaround is to ensure that the window is not clipped (for example, by raising the window to the top of the stack).

2.8.7 Setting an Application's Visual Class

Some X11 applications that run on an 8-bit X server may not run properly on a DECstation 5000 with a 24-bit frame buffer. If the application crashes with a `BadMatch` protocol error, the problem may be related to the way the application sets its Visual class. Many applications simply select the default Visual with the `DefaultVisual` macro.

The default Visual class on 8-plane systems is `PseudoColor`. The default Visual class on 24-plane systems is `TrueColor`. `TrueColor` uses a read-only, statically allocated, direct color map.

One quick workaround is to change the default visual class with the `-class` option on the `Xtm`(8X) command line to `PseudoColor.` However, it is likely you will also need to modify some applications to ensure that they select the appropriate visual on every X server. There is a good tutorial that describes how to approach this problem entitled ''Visualizing X11 Clients'' by David Lemke and David Rosenthal. It is available in the `doc/tutorials/visuals` directory on the X11 R3 and R4 release tapes. Section 7.7 of the ''Xlib Programming Manual'' by Adrian Nye (O'Reilly and Associates, Sebastopol, CA) also contains some explanation of this problem.

2.8.8 PostScript Previewer

Due to a lack of server resources, the PostScript Previewer's normal mode of operation is not supported by the PX, PXG, or PXG-turbo hardware configurations. The following procedure can be used to preview files on these systems:

1. Invoke the previewer without specifying a PostScript file on the command line.

   ```
   # dxpsview
   ```

2. Select the Watch Progress option from the Options pull-down menu. This tells the Previewer to interpret the PostScript commands directly into the window (normally it draws to a pixmap, and copies the image to the window).
3. Select the file to preview using commands from the File pull-down menu.

2.8.9 Display PostScript

Halftoning in Display PostScript does not work correctly on PXG-turbo models and 8-bit color PXG models.

2.8.10 Image Text

Prior to using image text, the fill-style in the current graphics context must be set to `FillSolid`. In a future release, image text will ignore the fill-style specified in the graphics context and use a solid fill-style (the normal procedure for image text described in the Xlib documentation).

2.8.11 Exceeding the Per-Process Virtual Size Limit in the DECstation/DECsystem 5000 Model 200PXG Server

Opening multiple double-buffered and/or Z-buffered windows may crash the server. For each double buffer (or Z-buffer) of a window, the server allocates virtual memory adequate to hold the buffer's contents, so that the associated VRAM can be freed for other uses (such as pixmaps).

The amount of virtual memory required for several large buffered windows can be prohibitive. A full-screen Z-buffered window, for example, requires 4.8 megabytes of memory. A double-buffered version of the same window requires twice this amount (4.8 megabytes for each buffer). The default storage limit is 65 megabytes (text segment + data segment + stack size) per process, so creating eight such windows will cause the `sbrk()` system call to fail, crashing the server.

To avoid the problem, raise the per-process data segment size limit, defined in the kernel configuration file by `MAXDSIZ`.

2.8.12 Off-Screen Memory Limitations Involving Large Pixmaps

Simultaneous use of multiple large pixmaps results in slowed performance.

2.9 DECsystem 5100

The following notes apply to the DECsystem 5100.

2.9.1 Installation Instructions for the DECsystem 5100

The installation instructions for the DECstation 3100 in the *Basic Installation Guide* and the *Advanced Installation Guide* support installations of the DECsystem 5100, except that the DECsystem 5100 has a new show device command.

To see what devices are configured on the DECsystem 5100, type the following command at the console:

```
>> conf
```

2.9.2 Default Boot Path

If the boot command is mistyped, the default boot path is used to boot the system. For example, to specify a boot path other than the default boot path, type the `boot` command with the `-f` option as follows:

```
>>  boot -f rz(0,1,0)genvmunix
```

However, if you mistype the preceding `boot` command, omitting the `-f`, instead of failing, the system will still boot based on the environment variable bootpath.

2.9.3 Backplate Labeling

The factory labeling of the console and terminal ports on the backplate of the DECsystem 5100 does not refer to the device major and minor numbers of the device special files made at installation time.

Please note that the device special file names may be changed after installation by the system manager, but the major and minor numbers will always be tied to the hardware line number.

Figure 2-3 shows how each console and terminal port corresponds to the device major and minor numbers and the device special files made at installation time.

Figure 2-3: DECsystem 5100 Console and Terminal Ports

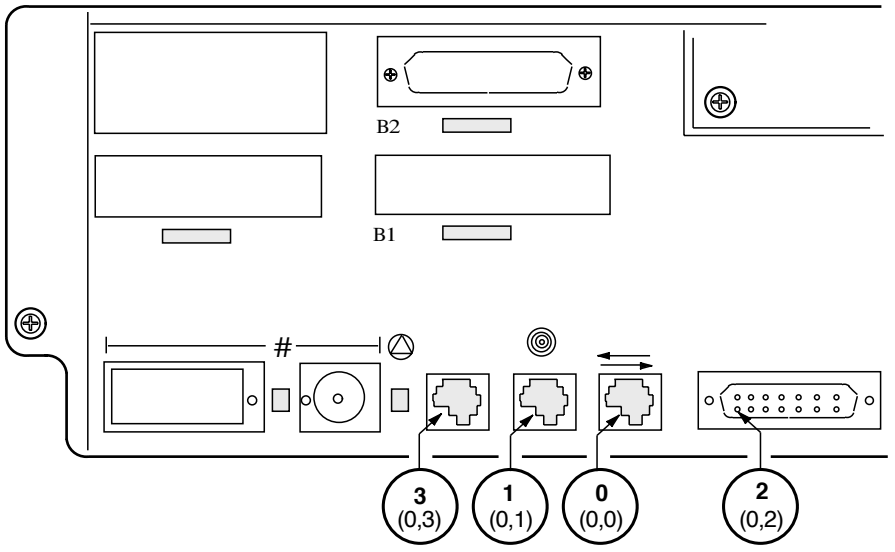


Table 2-3 shows the correspondence between the labels on the backplate of your DECsystem 5100 and the corresponding device major and minor numbers of the device special files made at installation time.

Table 2-3: DECsystem 5100 Console and Terminal Ports

Label	Installed ULTRIX Device Name	Major/Minor Number
0	/dev/console	0,0
1	/dev/tty00	0,1
2	/dev/tty01	0,2 (modem support)
3	/dev/tty02	0,3

Figure 2-4 shows how each console and terminal port corresponds to the device major and minor numbers and the device special files with the KN230 asynchronous communication option card added.

Figure 2-4: DECsystem 5100 KN230 with Async Terminal Ports

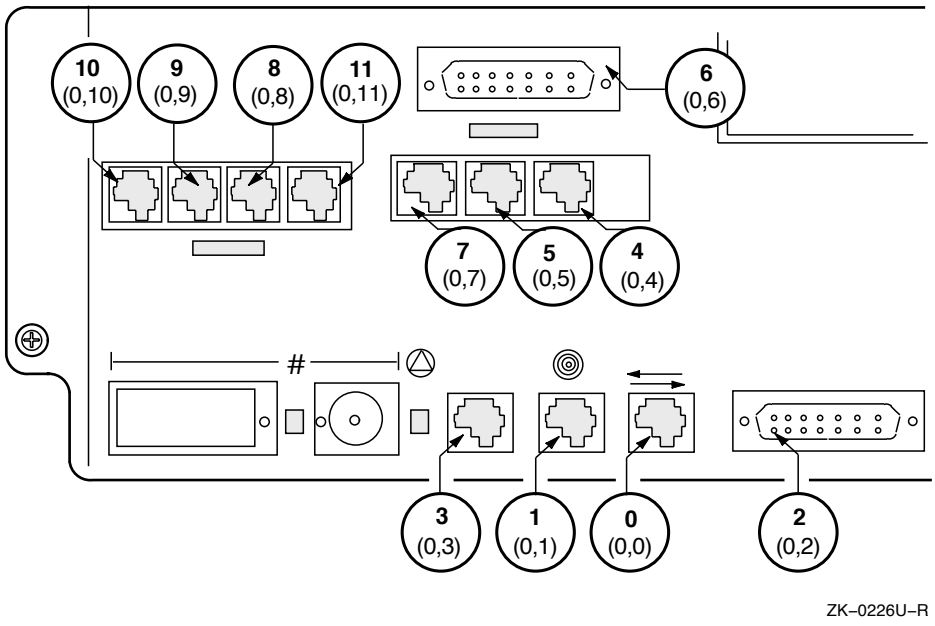


Table 2-4 shows the labels on the backplate of your DECsystem 5100 and the corresponding device major and minor numbers of the device special files if you have added the KN230 asynchronous communication option card.

Table 2-4: DECsystem 5100 KN230 with Async Terminal Ports

Label	KN230 ULTRIX Device Name	Major/Minor Number
0	/dev/console	0,0
1	/dev/tty00	0,1

Table 2-4: (continued)

Label	KN230 ULTRIX Device Name	Major/Minor Number
2	/dev/tty01	0,2 (modem support)
3	/dev/tty02	0,3
4	/dev/tty03	0,4
5	/dev/tty04	0,5
6	/dev/tty05	0,6 (modem support)
7	/dev/tty06	0,7
8	/dev/tty07	0,8
9	/dev/tty08	0,9
10	/dev/tty09	0,10
11	/dev/tty10	0,11

2.9.4 Configuring Terminal Devices for the KN230 Asynchronous Communications Card

To configure terminal devices for the KN230 asynchronous communications card, follow these steps:

1. Log in as `root` or become superuser.
2. Rebuild the kernel to add support for the two new devices which reside on the KN230 asynchronous option card. To rebuild your kernel, type the following command, replacing the italic *HOSTNAME* with the name of your system typed in all capital letters:

   ```
   # /etc/doconfig -c HOSTNAME
   ```

   The `-c` option specifies that the new kernel be built using the configuration file that already exists in the `/sys/conf/mips` directory.

   The `doconfig` command allows you to edit the configuration file. The following prompt appears immediately after you invoke the `doconfig` command with the `-c` option:

   ```
   Do you want to edit the configuration file (y/n) [n]?
   ```

   Answer yes to this prompt.

3. The `doconfig` program then places you in the `ed` editor. When you are in the `ed` editor, type the following sequence of commands to add the necessary lines to the `configuration` file:

   ```
   /mdc0
   a
   device          mdc1 at ibus?           vector mdcintr
   device          mdc2 at ibus?           vector mdcintr
   .
   w
   q
   ```

4. After you exit the `ed` editor, the `doconfig` program displays the following message as it begins to rebuild your kernel:

```
*** PERFORMING SYSTEM CONFIGURATION ***
.
.
.
```

When the `doconfig` program finishes, it reports the location of the newly built kernel as follows:

```
The new kernel is /sys/MIPS/HOSTNAME/vmunix
```

5. Move the new kernel to the `root` directory. Type the following command, replacing the italic *HOSTNAME* with the name of your processor in all capital letters:

```
# mv /sys/MIPS/HOSTNAME/vmunix /vmunix
```

6. The terminal devices are activated when you reboot your system using the new kernel. To reboot your system, use the `shutdown` command with the `-r` option and alert your users that the system will be going down, as follows:

```
# /etc/shutdown -r +30 "Rebooting to configure additional terminal lines"
```

For more information on shutting down your system, see `shutdown(8)`.

2.9.5 Adding Support for a New Option Card

The DECsystem 5100 CPU board (kn230 cpu board) provides support for one option card with two interrupts available for the expansion option. This allows the expansion option to have a maximum of two devices.

The Option ID Number is read from the OID (Option ID) register at startup time. This value should correspond to the value stored in the `iooption` console environment variable. To check or set the Option ID number, use the console environment commands, as follows:

```
>> printenv iooption
```

You must make an entry for each device in the kn230_option table in the file `/sys/data/kn230_option_data.c`. This provides a means of mapping the information needed to configure the devices to the ID value in the OID register. The first two entries in the kn230_option table of the `/sys/data/kn230_option_data.c` file are for the kn230 async card. You can use the following fields in these entries as guidlelines when adding another option card. The fields in the kn230_option table that you need to use when adding another option card are as follows:

option ID number	The value which will be read out of the Option ID register for the option card.
driver name	The device driver name as it appears in the system configuration file.

type	The type is either "D" if the device driver uses a device uba structure or "C" if the device driver uses a controller uba structure. No other values are allowed.
expansion0 csr	The csr address of the device which will interrupt on the first line. If there is only one device on the card, it should use the first available interrupt and you should place its csr address in the expansion 0 field.
expansion1 csr	The csr address of the device which will interrupt on the second line.
option_name	The option name string printed at boot time.

Note

Only one expansion csr field should be used for each entry; the other expansion card field should always be zero.

For example, assume that you are adding a new device driver called "new," which will be using only one interrupt. Since the driver will be using only one interrupt, you will need to make only one entry for the option card. In the following example, assume that the new option card has an ID number of 0x2. To add the device driver named "new," you would edit the kn230_option table in the file `/sys/data/kn230_option_data.c` as follows:

```
struct kn230_option kn230_option [] =
{

/* option driver       expansion 0  expansion 1        option name    */
/*  id#   name   type  csr address csr address           string      */
/* ====== ====== ====  =========== ===========  ====================  */

{  0x1,    "mdc",  'D',  0x15000000, 0x0, "Async comm (8 ports)"  },
{  0x1,    "mdc",  'D',  0x0, 0x15200000, "Async comm (8 ports)"  },

/* add additional option card devices here */

{  0x2,  "new",  'D',  0x15000000, 0x0, "New option card"       },

/*
 * DO NOT DELETE the Null entry which marks the end of the table
 * or your system will not configure properly.
 */
  {  -1,  "",  '0', 0,        0,   "0   }

};
```

2.9.6 Using the Halt Button on the DECsystem 5100

You can use the halt button on the back of the DECsystem 5100 to interrupt the ULTRIX operating system for the purpose of debugging the system.

However, to enable halts on the DECsystem 5100, you must first set the bootmode at the console level to halt. The default bootmode is set to reset, which will reset the system and run the diagnostics.

To set the bootmode at the console level to enable halts, enter the following command at the console:

```
>> setenv bootmode h
```

After the ULTRIX operating system boots with halts enabled, you can press the halt button to get to the console level where system status can be debugged and evaluated. To return to the ULTRIX operating system at the point where you pressed the halt button, use the `continue` command at the console as follows:

```
>> continue
```

Using the halt interrupt also affects the remaining three asynchronous terminal ports on the CPU board, since once a halt interrupt is issued, the DECsystem 5100 firmware is unable to restore the state of the remaining three terminal lines. Any action taking place on the remaining three terminal lines is therefore suspended.

To restore the three terminal lines, kill the `getty` process after you issue the `continue` command at the console and the system has returned you to single user mode, as follows:

```
# kill -1 1
```

Note

Due to the hardware and firmware constraints of the system architecture, the halt interrupt cannot be guaranteed to interrupt a system hang and bring you back to the console level.

2.9.7 Characters Output to Terminal Line Connections on Power Up

When you power on the DECsystem 5100 and have terminals or printers attached to the asychronous terminal ports, the hardware diagnostics output a series of characters to the attached terminals or printers. This problem does not affect the console port.

2.10 DECsystem 5400

These notes apply to the DECsystem 5400.

2.10.1 Possible Segmentation Faults During System Use

The following situations may cause a segmentation fault on your system:

- Using the `eyacc` command with no arguments. For more information on this command, see the `eyacc`(1) reference page.
- Using the `lookbib` command with no arguments. For more information on this command, see the `lookbib`(1) reference page.
- Specifying a network with the `rdate` program. For more information on the `rdate` program, see the `rdate`(8c) reference page.

2.10.2 Possible Performance Problem During N-Buffered I/O Use

If you use n-buffered I/O and the data buffers involved in the I/O are not properly aligned, you might experience a performance problem. You should consider the page size of the underlying system architecture when you set up your data buffers. If you fail to consider the underlying page size, the effect of using n-buffered I/O is negated.

To avoid this performance problem, you should align all user buffers that are involved in n-buffered I/O on system page boundaries. On both a DECsystem 5810 and a DECsystem 5400, a system page boundary is 4096 bytes.

The following program fragment demonstrates using the `getpagesize` system call to obtain the system page size. The fragment uses the result to align the buffer for optimum performance.

```
int pgsize;
int bufsize = 512;
char *cp;

pgsize = getpagesize();
cp = (char *)sbrk( bufsize + pgsize )
cp = (char *)((unsigned)((unsigned)cp + pgsize ) & ~( pgsize - 1 ));
```

For applications that use `malloc(3)`, `valloc(3)` can be used as a direct substitute, as follows:

```
#include <stdlib.h>

extern size_t  bufsize;
char      *bufp;

bufp = (char *)valloc(bufsize);
```

For more information on allocating size bytes aligned on a page boundary, see the `valloc(3)` reference page.

2.10.3 Server Logs Out During Daemon Startup

The `tftpd` server may log out of the root session when you start the daemon. You may want to avoid using the `tftpd` server.

2.10.4 Forcing a Crash Dump on the DECsystem 5400

If your system hangs for any reason, you can interrupt the operating system and force a crash dump. To interrupt the operating system, set the FUNCTION switch on the CPU cover panel to the up (dot inside the circle) position. Then, press the BREAK key, which sends an interrupt signal to the operating system. When the operating system processes the interrupt signal, it transfers control to the console program.

Once control transfers to the console program, the following appears on the display (the values on your system may be different from those shown here):

```
HALT PC 800dc968
Memory Size: 33554432 (0x2000000) bytes
Ethernet Address: 08-00-2b-0f-8e-42
>>
```

When the console prompt appears, issue the following command to force a crash dump:

```
>> go 0x80030008
```

In response to this command, the ULTRIX operating system performs a crash dump. Once the crash dump completes, you can reboot your system. However, be sure to set the FUNCTION switch to the down (no dot inside the circle) position before resuming the normal operation of your system. Failure to set the switch properly may result in accidental system interruptions.

2.10.5 Performance of dump(8) on DECserver 5800 and 5400 Series Processors

The `dump(8)` utility exhibits poor performance when dumping file systems on DECserver 5800 and 5400 series processors. The following workaround gives significant performance improvement to the utility.

To work around the problem, follow these instructions.

Note

Backups previously created with dump must be restored with the restore command; tapes created with the `dump`| workaround specified below must be restored with the `restore`| workaround, also specified below. Please note the date after which dumps are taken with the workaround procedure and/or mark tapes accordingly, so that the proper restore mechanism can be utilized.

The workaround is to use `dump` within a command pipeline with `dd(1)` to increase the performance significantly. Using a command like the following produces a dump tape which can then be read using `dd` and the `restore` utility:

```
# dump 0uf - / | dd of=/dev/rmt0h obs=10k
```

To restore the files from a tape created in this manner, use the following pipelined command:

```
# dd if=/dev/rmt0h bs=10k | restore xf -
```

Please note the following:

- The **obs=10k** option in the case of `dump` and the **bs=10k** option in the case of `restore` must be used.
- If using a TA90 tape drive or other drive equipped with a tape loader, the loader operations will not be automatic; the loading of tapes during the dump/restore must be performed by the operator/user.
- Other command line options for `dump/restore` can be utilized; those here are shown for example purposes.

Note

We recommend using **dd | restore** to read tapes created by the **dump | dd** command, specifically in the case where a dump image may span multiple volumes. The dump header and continuation information must be preserved across volumes when using dump and restore directly. This information is not written to tape when using the **dd** pipeline workaround. As a result `restore` can fail to read multivolume dump sets if not prefaced by the `dd` command.

Because of the small performance gain on a DECserver 5400 system with a TK70 tape drive and 3100-class processors, we do not recommend the dump workaround for these systems.

2.11 DECsystem 5500

The following notes apply to the DECsystem 5500.

2.11.1 Installation Instructions for the DECsystem 5500

The installation instructions for the DECsystem 5000, Model 200 Series Processor in the *Basic Installation Guide* and the *Advanced Installation Guide* support installations of the DECsystem 5500 with the following exceptions:

- There are new boot commands to support booting from a SCCI tape and a QBUS tape, respectively.
- Booting from a CDROM Optical Disk is supported on the DECsystem 5500.

2.11.1.1 The boot Command for SCSI Tapes – Use a command with the following syntax to boot your DECsystem 5500 from a SCSI tape:

boot -f tz(0, *unit-number*)

Replace the italic *unit-number* with the unit number of your tape drive.

The following example shows the command to boot the system from tape drive number 0:

```
>> boot -f tz(0,0)
```

2.11.1.2 The boot Command for QBUS Tapes – Use a command with the following syntax to boot your DECsystem 5500 from a QBUS tape:

boot -f tm(0, *unit-number*)

Replace the italic *unit-number* with the unit number of your tape drive.

Use the following command to boot your DECsystem 5500 from tape drive number 0:

```
>> boot -f tm(0,0)
```

2.11.1.3 Boot Command for the CDROM Optical Disc Kit – If your CDROM optical disk is not already in its caddy, follow the instructions in the hardware manual for inserting the optical disk into the caddy.

Follow this procedure to boot the system:

1. Load the CDROM optical disk labeled ULTRIX/UWS SUPP/UNSUPP (RISC) into the drive. Wait for the drive to be on line and ready.
2. Use the following command to determine the unit number of the drive for your device:

   ```
   >> show devices
   ```

   A display appears that shows what is assigned to each unit number on your system.

3. Use a command with the following syntax to boot your system:

   boot -f rz(*controller-number, unit-number*,0)vmunix

   Replace *controller-number* with the controller number. Replace *unit-number* with the unit number of your RRD40 optical disk drive.

   The following example shows the command to boot the system from RRD40 optical disk drive number 4:

   ```
   >> boot -f rz(0,4,0)vmunix
   ```

   Next, the installation software displays some system information, followed by the memory and hardware configurations.

2.11.1.4 Default Boot Path – If the boot command is mistyped, the default boot path is used to boot the system. For example, to specify a boot path other than the default boot path, you would type the `boot` command with the `-f` option as follows:

```
>> boot -f rz(0,1,0)genvmunix
```

However, if you mistype the preceding `boot` command, omitting the `-f`, instead of failing, the system will still boot based on the environment variable bootpath.

2.12 DECsystem 5800

The following notes apply to the DECsystem 5800 processor.

2.12.1 Possible Segmentation Faults During System Use

The following situations may cause a segmentation fault on your system:

- Using the `eyacc` command with no arguments. For more information on this command, see the `eyacc`(1) reference page.
- Using the `lookbib` command with no arguments. For more information on this command, see the `lookbib`(1) reference page.
- Specifying a database with the `rdate` program. For more information on the `rdate` program, see the `rdate`(8c) reference page.

2.12.2 Possible Performance Problem During N-Buffered I/O Use

If you use n-buffered I/O and the data buffers involved in the I/O are not properly aligned, you might experience a performance problem. You should consider the page size of the underlying system architecture when you set up your data buffers. If you fail to consider the underlying page size, the effect of using n-buffered I/O is negated.

To avoid this performance problem, you should align all user buffers that involved in n-buffered I/O on system page boundaries. On both a DECsystem 5810 and a DECsystem 5400, a system page boundary is 4096 bytes.

The following program fragment demonstrates using the `getpagesize` system call to obtain the system page size. The fragment uses the result to align the buffer for optimum performance.

```
int pgsize;
int bufsize = 512;
char *cp;
```

```
pgsize = getpagesize();
cp = (char *)sbrk( bufsize + pgsize )
cp = (char *)((unsigned)((unsigned)cp + pgsize ) & ~( pgsize - 1 ));
```

For applications that use `malloc(3)`, `valloc(3)` can be used as a direct substitute, as follows:

```
#include <stdlib.h>

extern size_t  bufsize;
char      *bufp;

bufp = (char *)valloc(bufsize);
```

For more information on allocating size bytes aligned on a page boundary, see the `valloc(3)` reference page.

2.12.3 Server Logs Out During Daemon Startup

The `tftpd` server may log out of the root session when you start the daemon. You may want to avoid using the `tftpd` server.

2.12.4 Interrupting the Operating System on a DECsystem 5800 Series Processor

If your system hangs for any reason, you can interrupt the ULTRIX operating system when it is running on a DECsystem 5810. To interrupt the operating system, set the top key switch to the ENABLE position. Then, press CTRL/P, which sends an interrupt signal to the operating system. When the ULTRIX operating system processes the interrupt signal, it transfers control to the console program.

Once control transfers to the console program, type the following at the console prompt to force a crash dump:

```
>> d %ra 0x80030008
%ra:    0x80030008          - 2147287032  '\b'
>> continue
dumping to dev 909, offset 65548
Dump of 8190 pages
.
.
.
```

Once the crash dump completes, you can reboot your system. However, be sure to set the Key switch to the SECURE position before resuming the normal operation of your system. Failure to set the switch properly may result in accidental system interruptions.

If you interrupt your ULTRIX system and immediately decide to return control from the console to the ULTRIX system, issue the `continue` command. You must issue this command before you issue any other command at the console prompt. If you issue a command other than the `continue` command, you must reboot your system to return control to the ULTRIX system.

Communication with the console terminal may stop after you issue the `continue` command. If communication stops, press the RETURN key. Pressing the RETURN key reestablishes communication and allows your system to operate normally.

2.12.5 Performance of dump(8) on DECserver 5800 and 5400 Series Processors

The `dump(8)` utility exhibits poor performance when dumping file systems on DECserver 5800 and 5400 series processors. The following workaround gives significant performance improvement to the utility.

To work around the problem, follow these instructions.

Note

Backups previously created with dump must be restored with the restore command; tapes created with the `dump`| workaround specified below must be restored with the `restore`| workaround, also specified below. Please note the date after which dumps are taken with the workaround procedure and/or mark tapes accordingly, so that the proper restore mechanism can be utilized.

The workaround is to use dump within a command pipeline with `dd(1)` to increase the performance significantly. Using a command like the following produces a dump tape which can then be read using `dd` and the `restore` utility:

```
# dump 0uf - / | dd of=/dev/rmt0h obs=10k
```

produces a dump tape which can then be read using `dd` and the `restore` utility.

To restore the files from a tape created in this manner, use the following pipelined command:

```
# dd if=/dev/rmt0h bs=10k | restore xf -
```

Please note the following:

- The **obs=10k** option in the case of `dump` and the **bs=10k** option in the case of `restore` must be used.
- If using a TA90 tape drive or other drive equipped with a tape loader, the loader operations will not be automatic; the loading of tapes during the dump/restore must be performed by the operator/user.
- Other command line options for `dump/restore` can be utilized; those here are shown for example purposes.

Note

We recommend using **dd | restore** to read tapes created by the **dump | dd** command, specifically in the case where a dump image may span multiple volumes. The dump header and continuation information must be preserved across volumes when using dump and restore directly. This information is not written to tape when using the **dd** pipeline workaround. As a result `restore` can fail to read multivolume dump sets if not prefaced by the `dd` command.

Because of the small performance gain on a DECserver 5400 system with a TK70 tape drive and 3100-class processors, we do not recommend the dump workaround for these systems.

ULTRIX Software Notes 3

This chapter discusses issues and known problems with the software and, when possible, provides solutions or workarounds to the problems.

The notes in this chapter cover the following topics:

- User Commands
- Administrative Commands
- System Calls
- Library Routines
- DECrpc
- Mail
- Network and Communications
- Printing
- Software Development
- ULTRIX/SQL
- VAX C

3.1 User Commands

This section discusses issues and known problems with user commands.

3.1.1 The ar(1) Command

When used to extract all files from an archive, `ar`(1) creates a file named `_____________ELEL__` with permissions 000 (it is a symbol table that is automatically created by `ar`). If a second `ar` command is run with the `x` key, `ar` displays the following message:

```
ar: Error: _____________ELEL__cannot create
```

You can ignore this message. You can avoid receiving this message by deleting the `_____________ELEL__` file.

3.1.2 The cp(1) Command

Issuing the `cp`(1) command using the **–r** flag to two directories with the same name causes infinite recursion and fills the file system or exceeds the user's quota limit.

3.1.3 Alias Causes csh(1) to Dump Core

The following alias causes csh(1) to dump core:

```
alias xxx 'foreach x ( 1 )\
anything'
```

Any use of the alias xxx causes csh(1) to dump core.

To avoid this problem do not use a backslash (\) when aliasing a foreach or while loop. A second way to avoid this problem is to use a C shell script instead of the alias.

3.1.4 The csh(1) Command Hangs on Double Quotes

In command line editing csh(1) will hang if a double quote (") is entered immediately after the initial escape character: <ESC-">.

3.1.5 The dd(1) Command

Do not specify a block size greater than 65535 bytes when performing device I/O with the TK50 device. This device will give an I/O error when a size greater than 65535 is given. If you attempt to give a block size larger than 65535 on other devices and the device returns an I/O error, try using the maximum size of 65535 bytes.

Also note that data corruption occurs when you move data from tape to disk files using the ibs=<block size> option with a block size greater than 1024. As a workaround, either use the bs=<block size> option, or set the number of write buffers to zero by using the wbuf=0 option.

Furthermore, when writing to tapes, you should not use an odd block size value when specifying the blocking factor. This can result in characters being dropped on every second block written to the tape.

Also, the buffering scheme employed by the VAX ULTRIX SCSI device driver limits the maximum tape record length to 16 Kbytes.

3.1.6 Caution on Using ln(1) Command

The behavior of the ln(1) command has changed. Previously, the -f option suppressed all but the usage message. If a file existed it would not remove the file and it would not produce an error message if -f had been specified.

In ULTRIX/UWS Version 4.1, if the file to be linked to already exists, the -f option removes the file if the permissions allow it, and creates a link. No error message is produced.

Also, the -i option causes ln(1) to solicit user response if the file to be linked to already exists. The -f option overrides this.

Sometimes, users specify the -f option in shell scripts to suppress the error messages. Now, -f removes the old link and makes a new one. (This is not a workaround, just a warning.)

3.1.7 The make(1) Command

Some cautions about using the make(1) command:

- If a makefile contains a dependency on a nonexistent file instead of always executing the rule, the current time is used for comparison. This can cause trouble when using make across an NFS environment if the time on the file server is later than the time on the client.
- The make(1) command treats two dollar signs ($$) as the end of a file name in a dependency list, not as a single dollar sign ($).

3.1.8 The sh(1) Command

The following notes apply to the sh(1) command.

3.1.8.1 Command Substitution Failure – Command substitution in the sh command fails to produce output when stdout is closed. The following sequence demonstrates the problem:

```
% /bin/sh
$ exec >&-
$ echo 'echo x' >&2
$
```

The echo command should print ''x,'' but nothing is printed. This example works correctly with sh5(1).

3.1.8.2 Version 7 Bourne Shell Not 8-bit Clean – The Version 7 Bourne Shell, sh(1), is not 8-bit clean. Only the System V Bourne Shell, sh5(1) is 8-bit clean.

If the ULTRIX Bourne shell, sh, is run from another program by the system or exec system calls whose maximum file descriptor in use is number 10, the prompt string is not printed. This can happen if a program has eight files open (in addition to the customary stdin, stdout, and stderr) at the time sh is called.

The problem does not occur with the System V Bourne shell, sh5, or with the C shell, csh. It also does not happen if file descriptor 11 is in use.

3.1.9 The size(1) Command Messages

The size(1) command on RISC machines can generate the following error messages:

```
ldopen: cannot read magic number filename
size: cannot open filename
ldinitheaders: magic number incorrect (0x0)
size: cannot open filename
```

These errors have the same meaning as the following VAX-based size message:

```
size: filename not an object file
```

3.1.10 The talk(1) Command Is Not 8-bit Clean

The talk(1) command is not 8-bit clean. Typing DEC Multinational Characters (ISO-8859/1) causes the characters to echo as a sequence of a caret (^) followed by the character represented with its high bit cleared.

This limitation makes talk(1) unusable if you want to communicate using a language which has ISO-8859/1 characters in its alphabet.

3.1.11 Using 8-bit Characters During telnet(1) or rlogin(1) Sessions

In order to use 8-bit characters during a telnet(1) or rlogin(1) session, the terminal from which you originate the session must be set to 8-bit mode. Likewise, if the /etc/gettytab file has not been set up properly for 8-bit mode, then you will not be able to make use of 8-bit characters. See the gettytab(5) reference page for more information.

Note that you must specify the -8 option to rlogin when using 8-bit characters, otherwise the connection will not be 8-bit clean. Also, if you cannot display 8-bit characters from the local terminal, you will not be able to display them from the remote terminal.

If you always want to use the -8 option, you can set up certain commands, depending on the shell you use.

For the csh shell, add the following command to your .cshrc file:

```
alias  rlogin  rlogin  *  -8
```

For the ksh or sh5 shells, add the following function definition to the .profile file:

```
rlogin()
     .
     .
     .
   { /usr/ucb/rlogin $* -8 }
```

Note for rlogin and telnet to work properly, only the local node needs to have pass8 set.

To check if 8-bit terminal mode is enabled, use stty -e and look for pass8. If you see pass8 , then 8-bit characters are enabled and your terminal is set up for 8-bit character support. If you see -pass8 then your terminal is set to 7-bit mode. In this case, you need to use stty pass8 to set the terminal to 8-bit mode. To check that your terminal is in 8-bit mode, enter a few 8-bit characters and check that they display properly.

In previous ULTRIX-32 releases, whether the pass8 flag to stty was enabled or not, the ULTRIX terminal driver would always pass 8-bit characters to the application. In the ULTRIX/UWS Version 4.1 release, the pass8 mode is now properly enforced by the terminal driver, so you must set this flag prior to using the 8-bit character set successfully.

3.1.12 The vi(1) Screen Editor

The command *n*d<CR> is supposed to delete *n* lines, but it deletes *n*+1 lines.

You can work around this problem by using the command *n*dd .

3.2 Administrative Commands

This section discusses issues and known problems with administrative commands.

3.2.1 The crash(8) Utility

The following notes apply to the crash dump facility.

3.2.1.1 Crash Dumps and the ps(1) Command – Using the `ps`(1) command on system crash dumps may not display the command arguments and the user environment.

The new system crash dump strategy does not dump user data for the default case of a partial dump strategy. The `ps`(1) command determines whether it is examining a system crash dump produced by the partial dump strategy. If this is true, the command makes no attempt to acquire the process' command line arguments or the process' environment strings.

The only workaround when the process argument strings and/or the process environment strings are required is to use the full dump strategy.

3.2.1.2 Dump Device Configuration Restrictions – The introduction of the new crash dump facility has restricted the type of dump device configurations permitted for the `rl, rk,` and `hp` devices. This restriction is identical to that for the `ra` type disks. For crash dumps to occur, the device that is specified in the host system configuration file as the dump device must be configured on the controller that will be used to boot the system device. Although this may have caused problems on a system due to limited disk space in the past, the new crash dump facility should eliminate this concern.

The facility now dumps just enough information for effective debugging of the system.

3.2.2 The fsck(8) Command

The following notes apply to the `fsck(8)` command.

3.2.2.1 Mounted File Systems and fsck(8) – In ULTRIX/UWS Version 4.1 the file system consistency checker, `fsck`, does not prevent the checking of mounted file systems. Running `fsck` on a mounted file system can cause the system to panic.

The `fsck` utility should only be invoked on unmounted file systems by the raw device. The only exception to this rule is the root (/) file system. The root file system is always mounted and should only be checked in single user mode by way of the block device when the system is booted.

3.2.2.2 Effects of New File System Timeout Algorithm on fsck(8) – The policy of changing the value of the clean byte to ensure checks of earlier file systems has been replaced in ULTRIX/UWS Version 4.1 by a file system timeout algorithm. This scheme limits the amount of time a file system is believed clean, no matter how the file system was mounted.

The timeout factor is initially set to 20 and is decremented when any one of the following three events occur:

- A file system is mounted
- 10,000 updates have occurred
- A file system was updated and `fsck` occurred more than 60 days earlier

When the timeout factor reaches zero, the following message is printed, and the next invocation of `fsck -p` will check the file system indicated in the message.

```
Warning, /dev/rxxx has exceeded %d %s threshold, fsck(8) is advised
```

where `%d` is replaced by the default factor, and `%s` is the event that crossed the threshold.

For example, if the timeout factor is 20, and a file system is mounted 20 times, the final mount will produce:

```
Warning, /dev/rxxx has exceeded 20 mount threshold, fsck(8) is advised
```

This message is a warning. The mount will succeed, and operations to the file system will continue. However, the invocation of `fsck -p` will check the file system.

The timeout factor can be set on a filesystem granularity. When a filesystem is made by `newfs(8)` or `mkfs(8)`, a default value of 20 is used. However, if timeouts occur too frequently or infrequently, the factor can be altered with `tunefs(8)`. Refer to `tunefs(8)` for more information about changing the value of the clean byte timeout factor.

Currently, when a ULTRIX-32 Version 3.1 or ULTRIX-32 Version 3.0 file system is mounted for the first time on a ULTRIX/UWS Version 4.1 system, this warning message is produced:

```
Warning, /dev/rxxx has exceeded 0 mount threshold, fsck(8) is advised.
```

This state continues until fsck is executed. If a mount is attempted of a file system used on a release prior to ULTRIX-32 Version 3.0, `fsck` will be mandatory, because the value of the clean byte was changed in ULTRIX-32 Version 3.0.

To view the value of the clean byte timeout factor, `dumpfs(8)` should be used.

3.2.3 License Management Facility (LMF)

The following notes apply to the ULTRIX License Management Facility (LMF).

3.2.3.1 License Management Facility Error – There is an error in the ULTRIX LMF utility which prevents the registration of any Product Authorization Key (PAK) containing a release date or a termination date that falls on the 30th or 31st of a month during a leap year. For example, a PAK containing the date 31-JUL-1992 will fail to register because of a checksum error.

There is no workaround to the problem. If you encounter the problem, request a replacement PAK from the issuer, whether it is Digital or a third party.

3.2.3.2 Error in lmfsetup(8) – Due to an error in the `lmfsetup(8)` PAK registration script, even if the PAK has been registered successfully, you must type CTRL/C to exit from the script.

3.2.4 The mkfs(8) Command (RISC Processors Only)

When `/bin/mkfs` is invoked without arguments, the RISC machines dump core instead of returning an error.

3.2.5 Changes to the rwhod(8) Command

The `/etc/rwhod` daemon is now commented out by default in the `/etc/rc` file. This impacts the `rwho(1c)` and `ruptime(1c)` commands. If you want to make this service available on your machine, you can enable the `rwhod` daemon by removing the comment (#) characters from the beginning of the `rwhod` lines.

3.2.6 Errors with tapex(8) Utility

The following notes apply to the `tapex(8)` utility.

3.2.6.1 Failures Using SCSI TZK10 Tape Drive – The `tapex` utility was originally written for variable length record tape devices. The TZK10 (QIC) tape drive uses fixed length records, which causes all variable length tests to fail. Many of the `tapex` tests can still be run with the TZK10, but tests such as the Random Record Size test, which you specify with the `-g` option, will fail with the TZK10.

In addition, the Append To Media test, which you specify with the `-d` option, also fails with the TZK10, because the TZK10 tape drive does not support data overwrite. When the Append To Media test fails, it generates the following errors:

```
Append to media testing.

This test simulates the behavior of the "tar r"
command by writing 20 records to the tape.
Next the tape is repositioned back one record and then
20 more records are written.
All records are of size 10240.
Finally the resulting tape read in for verification.

Aborting this test due write errors when trying to
append records to the media.
ERROR: 20 write errors occurred.
```

3.2.6.2 Record Size Validate Errors on DECsystem 5100 – The Record Size test fails on DECsystem 5100s with "validate errors" because the SCSI `sii` driver copies back more bytes than were actually read. In this test, the `tapex` utility initializes its read buffer to a known value, then attempts to read 1010 bytes to records of 1000 bytes. After the read, all bytes including the ten extra bytes are validated, causing the following validate errors:

```
Performing record size testing.  This test verifies that
at most one record is returned by a read system call.

Record size subtest #1:
Test read requests larger than the record size.
Request a read of 1010 bytes to records of size 1000 bytes.
The following errors were encountered when trying to
read more than a full record.  Read errors indicate
that more than a full record has been returned.
Read errors could also indicate that fewer bytes than
requested were returned.
FAILURE: 2000 validate errors
```

3.2.6.3 SCSI Command Timeout Failure – The Command Timeout test fails the forward skip file operation because the timeout is too short in the drivers of the SCSI host adapter. The Command Timeout test fails with the following error message:

```
MTIOCTOP failed, op = 1, count = 82
Failed MTIOCTOP command is MTFSF
The forward skip operation took 36 minutes.
FAILURE: unable to skip out 82 files
```

3.2.7 Layered Products and the setld(8) Command

Some layered products will not install because of an incompatibility with the `setld(8)` command. When you try to install them, these products will issue an error message and exit before the `setld` menu is presented. To install these products, set the environment variable `STL_NOACTM` to 1. On VAX and RISC systems, `csh(1)` users should enter the following command as `root`:

```
# setenv STL_NOACTM 1
```

Users of all other shells should enter this command:

```
# STL_NOACTM=1; export STL_NOACTM
```

Once you have set this variable, the product will install correctly.

Before installing any more products, unset the variable. On VAX and RISC systems, csh(1) users should enter the following command as `root`:

```
# unsetenv STL_NOACTM
```

Users of all other shells should log out of the system and log back in before installing more software.

3.2.8 The snmpsetup(8) Command Requires a Community Name

The `snmpsetup` command requires you to enter a community name even if communities are not be used in the network.

If you do not want to use communities in the network, enter a dummy community name when prompted by the `snmpsetup` command. At the completion of the `snmpsetup` command, delete the dummy community name from the `/etc/snmpd.conf` file.

3.2.9 System Exerciser and syscript(8)

Some of the system exercisers in `/usr/field` require their log files to be local to the exerciser. Because the `/usr` file system should be mounted read-only in a diskless environment, this prevents the creation of the clients' log file.

The workaround is to copy the desired exerciser to `/var/tmp` before executing it. This will move the exerciser and its associated log file into the clients' writable root area.

To run the `syscript` script, you should edit the file and globally change `/usr/field` to `/var/tmp/field`.

3.3 System Calls

This following notes apply to system calls.

3.3.1 The ptrace(2) System Call

Programs using `ptrace` to write into the instruction space of a traced program prevent that image file from being executed until the traced program has terminated. See `ptrace`(2), `dbx`(1), and `adb`(1) in the *ULTRIX Reference Pages*.

3.4 Library Routines

The following notes apply to library routines.

3.4.1 The execvp(3) Function

The `execvp` function will receive a SIGILL error if passed a PATH list that contains names greater than NAME_MAX (255) characters long.

3.4.2 The lint Library strncmp(3) Function

In the `lint` libraries, the third parameter to the `strncmp` function is incorrectly declared to be of type `int` instead of type `size_t`.

3.4.3 A printf(3) Problem (RISC Only)

On RISC systems, the `printf %f` format (and the `fcvt` function) incorrectly rounds down if all digits after the decimal point are zero. For example, with a %.1f format, the number .07 incorrectly prints as 0.0 instead of 0.1, but the number .17 prints as 0.2, the correct value.

3.4.4 Certain Comparison Routines Do Not Work with the qsort(3) Function

When you supply a comparison routine to `qsort` that can return different results for the same pair of keys at different times, `qsort` does not always execute properly. The `qsort` function may write beyond the bounds of the array being sorted.

The workaround to this problem is to provide a comparison routine that will consistently return the same result for any given pair of keys.

3.5 DECrpc

The following notes apply to DECrpc. The problems documented in these notes are temporary restrictions that will be corrected in a future release.

3.5.1 The NIDL Compiler Does Not Preserve Case Distinctions Correctly

The NIDL Compiler does not preserve case distinctions correctly. As a result, the `xx_cswtch.c` file generated from the `idl` file produces a `"syntax error"` message when you try to compile it. The problem occurs when mixed uppercase and lowercase characters are used to name the structure type, `ProducT` in this example:

```
%c
[ uuid(490da1ac7133.02.10.b5.c0.0c.00.00.00), version(1) ]
interface nidl_bug {

typedef struct {
     char type[81];
} ProducT;

int name$get_next (
     handle_t  [in] h,
     ProducT   [out]*product,
     char [out]product_name[81]);
}
```

The next example shows the `xx_cswtch` file generated by the NIDL Compiler. The NIDL Compiler generates the variable `ProducT` of type `ProducT`, instead of the variable `product` (all lowercase) as specified in the original `idl` file.

```
#define NIDL_GENERATED
#define NIDL_CSWTCH
#include "products.h"

ndr_$long_int name$get_next (h, ProducT, product_name)
handle_t h;
ProducT *ProducT;    /* The compiler gives an error on this.
                        The line should read ProducT *product */
ndr_$char product_name[81];
{
return (*nidl_bug$client_epv.name$get_next)(h, ProducT, product_name);
}
```

To avoid the compiler error, do not mix uppercase and lowercase characters in `idl` files. This is a temporary restriction in ULTRIX Version 4.0. The problem will be corrected in the next release.

3.5.2 Servers Generate an Error When Terminated with an Interrupt

A `longjmp botch` error is issued when an attempt is made to terminate a server with either CTRL/C or the `kill -INT [`*pid*`]` command. This error occurs because the NIDL Compiler does not insert a cleanup call in the `xx_sstub.c` code, such as that shown in the following example:

```
pfm_rls_cleanup (cleanup_rec, cleanup_status);
```

The problem only occurs if you use the `max_is` and `last_is` operations (or both) in the `idl` source file.

To avoid the problem, declare all arrays as fixed length in the `idl` file as illustrated in the following example:

```
int  [out]array[10]
```

3.5.3 The NIDL Compiler Does Not Generate Unique Names For Array Members

When the NIDL Compiler generates instructions to marshall and unmarshall members of an array, it does not provide unique local names in the output files. If you use a name that is the product of an array name concatenated with an underscore and an array member name, the C compiler generates errors when compiling the `xx_cstub.c` stub routine.

The idl file in the following example illustrates the cause of the errors:

```
%c
[ uuid(490da1ac7133.02.10.b5.c0.0c.00.00.00), version(1) ]
interface nidl_bug {

typedef struct {
     char name[81];
} array;

int name$get_next (
     handle_t  [in] h,
     array[out]*product,
     char [out]product_name[81]);
}
```

The NIDL Compiler defines the member of the product array as product_name, which conflicts with the user-defined string product_name[81]. When xx_cstub.c is compiled, the C compiler produces redeclaration errors like those shown in the following example:

```
"xx_cstub.c", line 57: redeclaration of xXx_b8a_
"xx_cstub.c", line 58: redeclaration of xXx_171e_
"xx_cstub.c", line 59: redeclaration of xXx_615_
```

To avoid the problem, do not define variable names that combine array names and array member name. For example, in the case in the previous example, change the product_name parameter to name_of_product or change the member name of the struct array to component_name[81].

3.5.4 The error_$c_text(3ncs) Routine Does Not Translate All nca_status Codes to ASCII Text

For some error codes, the error_$c_text(3ncs) routine generates messages like that shown in the following example:

```
status 1c000007 (network computing system)
```

It should generate a message such as the following:

```
invalid bound  (network computing system)
```

To determine the problem for untranslated error codes, use the grep(1) command to search the files /usr/include/idl/ncastat.idl and /usr/include/idl/fault.idl for the hexadecimal value. In generated error codes, the hexadecimal values a through f appear in lowercase. However, the hexadecimal constants for these error codes are defined with uppercase letters. This means that, although your program returns 1c000007, you must search for 1C000007.

3.5.5 Use of the max_is and last_is Attributes Produces Errors Across Hardware Architectures

Use of the max_is and last_is attributes in an idl file may produce errors across hardware architectures. Running a server on one hardware architecture (RISC) while running a client on another hardware architecture (VAX) sometimes produces invalid bound errors.

The errors occur because the NIDL Compiler does not insert a required instruction in the `xx_cstub.c` file. To avoid the problem, do not use the `max_is` and `last_is` attributes in applications that will run across multiple hardware architectures. Declare arrays of a fixed length in `idl` files.

3.5.6 The comm_status Parameter Must Be Declared As Both an Input and Output Parameter

When the `comm_status` parameter attribute is used as shown in the following example, the NIDL Compiler generates incorrect `xx_cstub.c` code and generates compiler errors:

```
%c
[uuid(4a9652fd445c.02.82.b4.06.5f.00.00.00), version(0)]

interface test
{
void test_bug(
     handle_t [in] h,
     status_$t [out, comm_status] *status);
}
```

The problem occurs because the NIDL Compiler does not see anything to be passed and omits required instructions. To avoid the problem, declare the `status` parameter as both an input and output parameter as shown in the following example:

```
status_$t [in,out, comm_status] *status
```

3.5.7 The lb_admin Utility Must be Restarted After Deletion of an Interface

The `lb_admin`(1ncs) utility loses track of the database after you delete an interface. If you follow a command to delete a registered interface with a command to list the remaining registrations, the list command fails. You must exit from the `lb_admin` utility and restart it in order to perform another function. The `lb_admin` utility incorrectly generates the messages in the following example (after a delete command followed by a list command) even if other interfaces are registered:

```
Data from GLB replica: ip:hostname

No entries match.
```

3.5.8 Bank Example Crashes with Illegal Instruction

In the following example of a `bank` application in the `/usr/examples/ncs/banks/README` file, the bank inquiry aborts the operation because it cannot find an active server for the "shawmut" bank:

```
bankd ip shawmut&                -- Start server listening on ip protocol

bank inquire shawmut Leach       -- Find out how much money Leach has

bank deposit shawmut Dineen 3    -- Give Dineen $3
```

The example `bank` application aborts the operation because it contains an imbedded call to the `abort()` function for certain error conditions.

To work around this problem, you can insert other error processing logic in place of the `abort()` routine.

3.5.9 Longjmp Botch Error

The `clean` command in `/etc/ncs/lb_admin` causes servers to experience a longjmp botch error on their next attempt to exit.

The `rrpc_$inq_interfaces()` routine used by the `lb_admin clean` command in `/etc/ncs/lb_admin` inserts a `pfm` handler on the stack at stub entry but does not remove it on exit. This causes the stack to be corrupted when the server attempts to exit.

This problem affects all active servers at the time the `clean` command is given. Servers corrupted by this problem must be killed manually with the `kill`-9 command and their registrations manually deleted from the location broker databases.

To work around this problem, avoid using the `clean` command to clear out unwanted registrations.

3.5.10 rpc_$bind Can Never Execute the rpc_$free_handle Call

The only status returned for `rpc_$bind` is `status_$ok`, even though things can go wrong in the routines called by `rpc_$bind`. As a result, `rpc_$bind` can never execute the `rpc_$free_handle` call and the application never knows something went wrong.

There is no workaround to this problem.

3.5.11 rrpc Routines Require an Explicit Call into the Entry Point Vector Table

On the client side, because of the way the `rrpc_` calls are defined and implemented in the `libnck.a` run-time library, you must explicitly call into the entry point vector table for the `rrpc_` interface to send an `rrpc_` request across the network. The following is an example of a call that works as desired:

```
(*rrpc_$client_epv.rrpc_$inq_interfaces)(handle,
           (unsigned long) max_ifs, ifs, &l_if, &status);
```

The server side stub routines call the entry point `rrpc_$inq_interfaces` on behalf of the client. The results of the call are then passed back to the client.

3.6 Mail

The following notes apply to the mail utility.

3.6.1 The sendmail Program Does Not Set the $x Macro on Received Mail

The `$x` macro, which represents the full personal name of sender, is not set by the `sendmail` program when the sender is not local. Users who receive mail through the DECnet-Internet Gateway do not see the personal name of the person who sent the message.

This behavior is due to the way mail headers are constructed and cannot be changed. The `sendmail` program uses information in the `/etc/passwd` file to get personal names; the program has no access to remote password files.

3.6.2 Creating Aliases That Exceed 1024 Characters in /usr/ucb/mail

The `/usr/ucb/mail` program limits private user aliases to 1024 characters.

The string that you are aliasing in your `.mailrc` file cannot exceed 1024 characters. Aliases defined in a `.mailrc` file that exceed this length limit cause the `mail` program to core dump through a segmentation fault.

To work around this problem, redefine the long alias using aliases that are less than 1024 characters each. The original alias can then be constructed using those smaller aliases.

```
alias a a1,a2,a3...aN (list 1 of less than 1024 characters)
alias b b1,b2,b3...bN (list 2 of less than 1024 characters)
alias c a,b (complete alias list)
```

For example, assume that you want to define an alias c that exceeds 1024 characters in length. First, break alias c into aliases a and b. Then define alias a to include list 1 and alias b to include list 2. Finally, define alias c to include the lists that are defined by aliases a and b.

3.6.3 sendmail Address Parsing Problem

There is a problem with the sendmail configuration file rules for parsing addresses of the form:

```
nnnn::uuuu@hhhh
```

If this address is interpreted according to the RFC822 specification, the canonical form is:

```
nnnn::uuuu<@hhhh>
```

This is often incorrect if the mail actually originated from `uuuu@hhhh` and passed via `nnnn` on the way to the receiver's host. This typically happens when two ULTRIX users communicate via the DECnet mail mailer because the recipient has a `.forward` file on host `nnnn` specifying a DECnet mail forwarding address.

If this is the case, then the required canonical form is:

```
uuuu@hhhh<@nnnn.enet>
```

Implementing this functionality violates the RFC822 specification. To help resolve this problem, the following rule is included in rule set 3:

```
R$-::$+<@$~S>      $1::$2@$3          defocus - not local host
```

This rule switches between the two forms of the canonical address depending on whether the host `hhhh` is known to be a local host (one listed in `/etc/hosts` or `/etc/hosts.local`) or not.

If `hhhh` is local, the canonical form is assumed to be:

```
nnnn::uuuu<@hhhh>
```

Otherwise it is:

```
uuuu@hhhh<@nnnn.enet>
```

This presumes that local users will use tcp mail (user@host) rather than using DECnet mail (host::user) to forward mail.

It is recommended that users forward mail using tcp local mail (user@host) when possible. If full adherence to RFC822 is required, then the preceding rule can be commented out of rule set 3.

3.6.4 sendmail Sender Name Problem

The `sendmail` program uses information in `/etc/utmp` to determine the contents for the `From:` line in outgoing mail that has originated locally.

The contents of `/etc/utmp` change frequently and `sendmail` can, occasionally, read inconsistent data. This results in bad `From:` line specifications on outgoing mail.

3.6.5 The Rand Mail Handler

Be aware of these problems:

- The `send` command does not recognize `–nowait` with "send: -nowait" in `.mh_profile`.
- The `send` command (with the `–ali` option specified and a nonexistent alias file encountered) exits, rather than ignoring the nonexistent alias file.

3.7 Network and Communications

The following notes apply to network and communications.

3.7.1 The ne Network Device

The `ne` (SGEC) device driver reports more packet collisions than the `ln` (LANCE) driver under the same network environment. The LANCE chip can only report one or at the most two collisions when transmitting a packet. The SGEC chip reports the real counter of the collisions, from 1 to 15.

As a result, the `netstat -i` command will report a higher collision rate when run on a `ne` network device.

3.7.2 Writing to a Remote a.out File

If a remote a.out image file is written to on a server while one or more clients are using that image file, further references on that file by the currently executing client processes will cause those processes to be killed. Under these conditions, the system responds with the message:

```
pid <number> killed due to text modification
```

The <*number*> argument is the pid number of the process that was killed.

If another process is started up on a given client while processes on that client are being killed, the new process will fail and the system responds by displaying the message:

```
remote text modified and not yet cleaned up
```

In this case, retry the process.

3.7.3 Nonexisting Pathnames in /etc/exports

Beware of nonexisting pathnames in the `/etc/exports` file. The mount daemon can dump core if such an entry exists with a long hosts/netgroups list.

If the `/etc/exports` file contains a pathname that does not exist and has a long list of hostnames or netgroups associated with it, the mount daemon can cause a segmentation fault. More specifically, the problem occurs if the nonexisting exports entry is more than one line long.

If the nonexisting pathname is the first listed in the exports file, the mount daemon will dump core; otherwise, portions of its hosts list will be associated with the previous exportable directory.

To work around the problem, do not export nonexisting directories.

3.7.4 Login and Security Restrictions

There are limitations with remotely served authorization (auth) databases.

The auth database cannot be served through YP, only by BIND/Hesiod. Do not attempt to serve the auth data base through YP. Instead, use `svcsetup(8)` and `secsetup(8)` to specify your services.

When auth entries are served through BIND/Hesiod, login fail count maintained by `login(1)` is not supported. Login attempts that fail on BIND clients will not increment the login fail count if the account is not on the local machine. Repeated login failures for any account are always recorded in the local system log.

When the security features are enabled, it is not possible to `su` to root on lines which are not marked as secure in the `/etc/ttys`file

3.7.5 Address Change for the Network Information Center (NIC)

This note is a recast of the DDN MGT Bulletin 72 of 06Apr90.

In order for the NIC to provide better service, and because of the phaseout of the ARPANET, the following changes have taken place and are effective immediately (6 April 1990).

NIC.DDN.MIL

The new address for host NIC.DDN.MIL is 192.67.67.20. The old ARPANET address for the NIC, 10.0.0.51, is no longer available, due to the phase-out of the ARPANET.

The old MILNET address for the NIC, 26.0.0.73, will continue to be valid until 1 June 1990, after which service to this address will be discontinued.

ROOT DOMAIN SERVER

The NIC's root domain server will run on a new host, NS.NIC.DDN.MIL at address 192.67.67.53. The old root server will continue to run on NIC.DDN.MIL until 1 June 1990.

New host tables and domain server files produced by the NIC will reflect the new addresses.

Users should update host tables, domain server boot files, manuals, and documentation to reflect the new addresses. Use these addresses to contact the NIC.

Table 3-1 shows the current list of root servers:

Table 3-1: Network Information Center Addresses

Root Server	Address
NS.NIC.DDN.MIL	192.67.67.53
A.ISI.EDU	26.3.0.103 128.9.0.107
AOS.BRL.MIL	128.20.1.2 192.5.25.82
C.NYSER.NET	192.33.4.12
GUNTER-ADAM.AF.MIL	26.1.0.13
NS.NASA.GOV	128.102.16.10 192.52.195.10
TERP.UMD.EDU	128.8.10.90

3.7.6 Maintaining the BIND/Hesiod Root Name Server Data File

There are currently seven BIND root name servers. These servers know about all the top-level BIND domains on the Internet network. It is necessary to know about these servers when making queries about hosts outside of your local BIND domain. Be aware that the host names and IP addresses of these machines do periodically change. It is imperative that these changes are reflected in the BIND/Hesiod root name server data file, `/var/dss/namedb/named.ca`. You must update your root name server file as soon as you install ULTRIX/UWS Version 4.1 to reflect the NIC's address change of April 6, 1990.

To maintain the file, at least once a month connect to the system nic.ddn.mil which has an IP address of 192.67.67.20, and is managed by the Network Information Center. Use the `ftp(1)` command with a login "anonymous" and password "guest". Retrieve the file, NETINFO:ROOT-SERVERS.TXT, and examine it against your existing `named.ca` file. If any differences exist, incorporate them into the existing format in the `named.ca` file.

For example, the following is the `ftp` session described above from host `chicago.cities.dec.com` with IP address 128.11.22.33 and user name jones:

```
chicago.cities.dec.com> ftp nic.ddn.mil.
Connected to nic.ddn.mil.
220 NIC.DDN.MIL FTP Server Process 5Z(47)-6 at Wed 11-Apr-90 08:24-PDT
Name (nic.ddn.mil.:jones): anonymous
Password (nic.ddn.mil.:anonymous):
331 ANONYMOUS user ok, send real ident as password.
230 User ANONYMOUS logged in at Wed 11-Apr-90 08:24-PDT, job 40.
ftp> get netinfo:root-servers.txt /tmp/root-servers
200 Port 11.175 at host 128.11.22.33 accepted.
150 ASCII retrieve of TS:<NETINFO>ROOT-SERVERS.TXT.18 (1 page) started.
```

```
226 Transfer completed. 673 (8) bytes transferred.
local: /tmp/root-servers remote: netinfo:root-servers.txt
673 bytes received in 0.09 seconds (7.3 Kbytes/s)
ftp> quit
221 QUIT command received. Goodbye.
chicago.cities.dec.com>
```

If you do change your `named.ca` file, you should keep another copy of the original file, that is `named.ca.nic`. You should do this because if the `bindsetup(8)` command is rerun, it will overwrite the `named.ca` file. If the `bindsetup` command is rerun, be sure to copy back the `named.ca.nic` file to `named.ca`.

If you are unable to `ftp` to `nic.ddn.mil`, try sending mail to

```
hostmaster@nic.ddn.mil
```

or calling the NIC's toll-free number, (800) 235-3155.

3.7.7 Automatic daemon startup on BIND/Hesiod Primary Server Using bindsetup

When the `bindsetup`(8) command is run for the first time to set up a BIND/Hesiod primary server which is distributing the passwd database, it adds the "bindmaster" alias to the hosts database in the directory, `/var/dss/namedb/src`. If you reply "yes", to the automatic daemon startup question, `bindsetup` tries to start up the `hesupd`(8) daemon. This daemon requires the ability to look up the "bindmaster", but it cannot since the "bind" has not yet been added on the hosts entry in the `/etc/svc.conf` file. The message "hesupd can't lookup bindmaster" will appear as the `bindsetup` command continues.

To work around the problem, you must give the primary server a way to lookup the "bindmaster" alias. You can do this two ways. Either you can add the "bindmaster" alias to the `/etc/hosts` entry for the primary server, or you can modify the `/etc/svc.conf` file to use "local,bind" for the hosts database. Then start up the `hesupd` daemon manually or reboot.

3.7.8 Using the Packetfilter with Multiple Ethernet Interfaces

When using the Packetfilter facility on multiinterface systems, all interfaces must have been configured at least once before the packet filter will recognize all interfaces. If all of your network interfaces are configured up and running, then there is no problem.

However, if your system has two network interfaces and the first interface has not been enabled with `/etc/ifconfig`, the second interface will not be seen by the packetfilter.

To work around this problem, you should enable all of the interfaces on your system. If an interface cannot be enabled for some reason, you can enable it using a dummy address, and then mark the interface "down". Choose an address which is not currently in use on your LAN.

For example, consider a MicroVAX-II that contains two DEQNA interfaces, but only the second interface, qe1, is currently in use. To allow the packetfilter access to qe1, the qe0 interface must first be configured with a dummy address and then marked "down":

```
# /etc/ifconfig qe0 1.0.0.1
# /etc/ifconfig qe0 down
```

3.7.9 Protecting YP and BIND/Hesiod Files and Directories

To protect your YP maps from malicious users, change the modes of your `/var/yp/<` *your Domain name* `>` directory to 700 (drwx------).

To protect your BIND/Hesiod databases specified in `named.boot` file on secondary servers from malicious users, change the modes of the database files in the `/var/dss/namedb` directory to 600 (-rw-------) .

To protect your YP and BIND/Hesiod source files on your YP master and BIND/Hesiod primary server, change the modes of your `/var/yp/src` and your `/var/dss/namedb/src` directories to 700 (drwx------).

3.7.10 Improve Your Yellow Pages Makefile

In the beginning of the Makefile are variable definitions. Due to a possible error in the `make` command, the following line does not expand properly:

```
DOM='domainname'
```

As a result, whether or not a YP source file has been changed, the map will always be updated.

To work around the problem, unnecessary map updates, modify the `/var/yp/Makefile` and substitute your YP domain name. For example, if your YP domain name is "yourYPdomainname," the new line looks like this:

```
DOM=yourYPdomainname
```

3.7.11 Recommendation For Placement of NFS Mount Points

It is recommended that you do not place NFS mount points to different servers in the same directory. If mount points to different servers are placed in the same directory and one of the servers is hard mounted and goes down, all NFS requests will hang until the server that is down comes back up.

When computing the pathname string of a directory, `getwd(3)` moves up the directory tree from the current working directory to the root. When `getwd` passes through a mount point, it will `stat(2)` entries in the directory until it finds the mount point it just traversed. If any of the entries in the directory are mount points to a server that is hard mounted and down, `getwd` will block until the server responds.

3.7.12 NFS Filesystems and Named Pipes Incompatibility

In ULTRIX/UWS Version 4.1, the over-the-wire NFS representation of a named pipe has been modified to follow the Sun representation introduced in SunOS Version 4.0, and in other NFS vendor's ports based on SunOS Version 4.0. As a result of this change, ULTRIX/UWS Version 4.1 is incompatible with prior ULTRIX releases when named pipes are being used in NFS filesystems.

If an `ls` command shows a character device where a named pipe was expected, then patches need to be installed.

Patches to upgrade a Version 3.1 system to be compatible with ULTRIX/UWS Version 4.0 or ULTRIX/UWS Version 4.1 (and other NFS vendors) are located in `/usr/mdec/named_pipe`. Both RISC and VAX patches are provided.

3.7.12.1 Sample Patch Procedure for a VAX Machine – In this example, the ULTRIX-32 Version 3.1 host is called "old," and the ULTRIX/UWS Version 4.1 host is called "new."

First, copy the files to "old":

```
old# rcp new:/usr/mdec/named_pipe/nfs_subr.o.31vax /tmp
old# rcp new:/usr/mdec/named_pipe/vnodeops_gfs.o.31vax /tmp
```

Next, as root, go to the kernel binary pool, and copy the files into it, saving the originals:

```
old# cd /sys/BINARY.vax
old# mv nfs_subr.o nfs_subr.o.save
old# mv vnodeops_gfs.o vnodeops_gfs.o.save
old# cp /tmp/nfs_subr.o.31vax nfs_subr.o
old# cp /tmp/vnodeops_gfs.o.31vax vnodeops_gfs.o
```

Then, build a new kernel:

```
old# cd ../APRIL
old# make
/bin/cc -I. -c -S -DAPRIL -DVAX3600 -DVAX3900 -DEMULFLT -DUFS -DRPC \
-DNFS -DINET -DKERNEL -DUPGRADE=0 -DSWAPTYPE=0 ../vax/swap.c
/lib/c2 swap.s | ../vax/inline/inline | as -o swap.o
rm -f swap.s
loading vmunix
rearranging symbols
text    data    bss     dec     hex
460556  91884   533304  1085744 109130
```

Then, copy the kernel into the root, saving the old one:

```
old# mv /vmunix /vmunix.orig
old# cp vmunix /vmunix
```

Finally, reboot the system:

```
old# /etc/reboot
```

3.7.12.2 Sample Patch Procedure for a RISC Machine – In this example, the ULTRIX-32 Version 3.1 host is called "old," and the ULTRIX/UWS Version 4.1 host is called "new."

First, copy the files to "old":

```
old# rcp new:/usr/mdec/named_pipe/nfs_subr.o.31mips /tmp
old# rcp new:/usr/mdec/named_pipe/vnodeops_gfs.o.31mips /tmp
```

Then, as root, go to the kernel binary pool, and copy the files into it, saving the originals:

```
old# cd /sys/b.mips/BINARY
old# mv nfs_subr.o nfs_subr.o.save
old# mv vnodeops_gfs.o vnodeops_gfs.o.save
old# cp /tmp/nfs_subr.o.31mips nfs_subr.o
old# cp /tmp/vnodeops_gfs.o.31mips vnodeops_gfs.o
```

Then, build a new kernel:

```
old# cd ../MARCH
old# make
/bin/rm -f a.out a.out.q assym.h
cc -DTIMEZONE=300 -DDST=1 -DMAXUSERS=32 -DMAXUPRC=50 -DPHYSMEM=8 \
-DNCPU=1 -DDMMIN=32 -DDMMAX=4096 -DBUFCACHE=10  -I. -I. \
-DMARCH -DR2000a -DUWS -DDLI -DLAT -DDECNET -DSYS_TRACE \
-DRPC -DUFS -DNFS -DINET -DQUOTA -DMIPS -DKERNEL -g ../machine
/genassym.c
/bin/rm -f a.out
/bin/rm -f entry.o
cc -EL -I. -c -G 8 -O0 -g3 -I. -DMARCH -DR2000a -DUWS -DDLI -DLAT \
-DDECNET -DSYS_TRACE -DRPC -DUFS -DNFS -DINET -DQUOTA \
-DMIPS -DKERNEL -DLOCORE -g -DTIMEZONE=300 -DDST=1 \
-DMAXUSERS=32 -DMAXUPRC=50 -DPHYSMEM=8 -DNCPU=1 -DDMMIN=32 \
-DDMMAX=4096 -DBUFCACHE=10  ../machine/entry.s
/bin/rm -f vmunix
/bin/sh ../conf/mips/newvers.sh
cc -EL -I. -c -G 8 -O2 -g3 -I. -DMARCH -DR2000a -DUWS -DDLI -DLAT \
-DDECNET -DSYS_TRACE -DRPC -DUFS -DNFS -DINET -DQUOTA \
-DMIPS -DKERNEL -g -c vers.c

uopt: Warning: file not optimized; use -g3 if both optimization \
and debug wanted
loading vmunix
echo ../mips/symbols.sort vmunix
/bin/size vmunix
text    data    bss     dec     hex
840112  166352  177856  1184320 121240
/bin/chmod 755 vmunix
```

Then, copy the kernel into the root, saving the old one:

```
old# mv /vmunix /vmunix.orig
old# cp vmunix /vmunix
```

Finally, reboot the system.

```
old# /etc/reboot
```

3.7.13 DLI Programs Must Be Recompiled

The `sockaddr` structure for DLI was inadvertently changed, due to another change in the `<net/if.h>` header file. Therefore, any DLI programs must be recompiled.

3.7.14 DLI/802 Passes Up Packets That Should Be Dropped

When a user opens a DLI/802 socket and enables an individual and/or group SAP, the socket can receive "Unnumbered Information" messages sent to that SAP with the poll bit set to 1. To detect this event, the user must use the `recvfrom` call and process the control field passed in the address/header structure. This is done by testing bit 4 (assuming the low order bit is numbered 0) of the `U_fmt` member of the `osi_802hdr` structure. "Unnumbered Information" packets should only have the poll bit set to 0. If the bit is set to 1, the packet should be ignored.

3.7.15 MOP Request Counters Function Does Not Work for VAX Systems with DEBNAs

The Data Link Interface (DLI) does not respond to MOP Request Counters messages if the message is received over a DEBNA.

This condition is unlikely to affect anyone but users trying to troubleshoot a network problem. The desired information can be obtained if DECnet is installed on both the requesting system and the target system by executing the following command:

```
# ncp tell node nodename show line bnt-0 counters
```

In this example, *nodename* is the name of the remote node from which the information is being solicited.

3.7.16 The snmpsetup Command Requires a Community Name

The `snmpsetup` command requires you to enter a community name even if communities are not to be used in the network.

If you do not want to use communities in the network, enter a dummy community name when prompted by the `snmpsetup` command. At the completion of the `snmpsetup` command, delete the dummy community name from the `/etc/snmpd.conf` file.

3.7.17 DEMNA Adapter Not in netsetup Script

The netsetup script, which allows system administrators to configure their local area network, fails to list the Digital DEMNA XMI bus to Ethernet Adapter as a supported network adapter for ULTRIX. The table of common network device names lists the following adapters:

```
Device Name     Description
-----------      -----------
   xna0          DEBNI - BI bus
   ni0           DEBNT, DEBNA - BI bus
   de0           DEUNA, DELUA - UNIBUS
   .
   .
   .
```

The table should read as follows:

```
Device Name     Description
-----------      -----------
   xna0          DEBNI - BI bus
   xna4          DEMNA - XMI bus
   ni0           DEBNT, DEBNA - BI bus
   de0           DEUNA, DELUA - UNIBUS
   .
   .
   .
```

When selecting the DEMNA as a networking interface, answer "xna4" as the device name:

```
"What is the device name of your Network Interface [xna4]?"
```

3.8 Printing

The following notes apply to printing.

3.8.1 The lpr(1) Command

The `lpr`(1) command allows control characters to be printed using the `–l` option. Even when this option is specified, print jobs are piped through the `of` filter, or whatever filter is specified in the submitting `lpr`(1) command. There are some control characters, for example `^A` and `^Y`, that have special meanings for the filters.

If the file you want to print contains these control characters, it may cause the filters to hang, in which case you will have to remove the job from the queue using the `lprm`(1) command. You may also have to change the control characters in the file. If you change the control characters in the file, use the `lpr`(1) command with the `–x` option. This is a transparent filter that allows all data to be passed to the printer unchanged.

3.8.2 Notes on lprsetup(8)

The following notes apply to the `lprsetup`(8) command.

3.8.2.1 Default Values Set by lprsetup – The current default value for the `xs` parameter is 040040. This value is incorrect. The value should be changed to 044000 (octal) for the following printers:

- la50
- la75
- la100
- la210

The following printers work with either value set:

- lcg01
- lj250
- ln03
- ln03s
- ln03r
- lg31

The default values set by `lprsetup` correspond to the factory settings for each printer with the exception of the ln03, ln03s, and lj250. For these printers the factory setting for baud rate is 4800 while the default for `lprsetup` is 9600. You must either change the entry in the `/etc/ttys` file to 4800 or you must change the printer speed to 9600.

3.8.2.2 lprsetup Command Defaults to No Parity – For known serial printers (with the exception of the LA50 and the LA75), the `lprsetup` command sets the `fs` flag in `/etc/printcap` to no parity. The `fs` flag for the LA50 and LA75 is set to even parity. Having no parity can cause problems especially if the printer's dip switches are set to a specific parity, because the terminal driver will set the line to some

default parity. This may or may not be the same as the printer's settings. Therefore, the printer's dip switches should be set to a known parity and the `fs` flag in `/etc/printcap` should be set explicitly to match the printer.

To do this, edit `/etc/printcap` or use the `lprsetup` command when creating the printing environment. See the `sg_flags` field in `tty`(4).

3.8.3 Printing Large Files

The default size limit of a file to be printed is 1025024 bytes (local or remote).

As superuser, you can modify the default by editing the `/etc/printcap` file. Insert the parameter `mx#0` or `mx#N` (where N is an integer) in the printer's `printcap` entry. Now the file size limit is either unlimited or N blocks of 1024 bytes, respectively. Do this on the local system (where you issue the `lpr` command). The limit applies both to local print jobs and to those print jobs sent to remote systems. The limit set at the local system does not affect print requests sent from remote systems.

As a nonprivileged user, you can work around the file size limit by dividing the file into sections less than 1025024 bytes before printing, assuming that the file data format allows for this. Or, you can use the `rcp` command to copy the file to a system with a printer that has a sufficiently large file size limit. Log in on that system and then request local printing.

3.8.4 Retrying Print Jobs Indefinitely

If a `/etc/printcap` entry does not specify the `:of:` or the `:if:` capability, it is possible that the line printer daemon `lpd` will retry print jobs indefinitely.

To prevent this, add the following to the `/etc/printcap` entry:

```
:if=cat%0:\
```

This specifies the `cat` program as the input filter, the `%0` switches off the passing of the default arguments for the input filter.

3.8.5 Spool Directories for Remote Printers

If you manually create a `printcap` entry for a remote printer, you must also create a spool directory for that printer, just as you would for a local printer. For example, if you have an LA100 printer, with a `printcap` entry of `:sd=/usr/spool/la100`, then you should type the following commands to create its spool directory with the proper owner:

```
# mkdir /usr/spool/la100
# chown daemon /usr/spool/la100
```

If you use the `lprsetup` script to add a printer, the spool directory is created for you.

3.8.6 PrintServer Client Software for ULTRIX

In Versions 3.0 and 3.1 of ULTRIX–32, print spooler support for the PrintServer family of network printers was supplied through a layered product, Version 2.0 of the PrintServer Client Software for ULTRIX.

In ULTRIX/UWS Version 4.1, the print spooler enhancements are part of the operating system, so you must not install the PrintServer Client layered product.

The print spooler provides network printing services on the PrintServer 20, the PrintServer 40, and the PrintServer 40 Plus network printers over a DECnet or TCP/IP network.

For management, booting, and file services, you still need to install either the PrintServer Supporting Host for ULTRIX (to boot and manage from ULTRIX) or the VAX PrintServer Supporting Host (to boot and manage from VMS).

For PrintServer network printers on a DECnet network, you require the VMS PrintServer Supporting Host Software, Version 2.0, 2.1, or 3.0, running on a VAX system.

For PrintServer network printers on a TCP/IP network, you require the PrintServer TCP/IP Supporting Host Software, Version 1.0 or 2.0, running on an ULTRIX system.

Each client, supporting host, and PrintServer printer is either a DECnet or TCP/IP node on the network.

You can print PostScript, ANSI, ASCII, ReGIS and TEK4010/4014 files from an ULTRIX system on a PrintServer located on the same local or wide area network. On a DECstation 3100 you can print ASCII, ANSI, and PostScript files only.

3.8.6.1 The lpr –D Option – If the `lpr –D` *(data_type)* option is misspelled, the job fails and the message "translator not found" is reported in the error log. You are not notified of the job failure.

The installation default data type is ASCII, so if you do not specify a data type with the `lpr` command, or in the `printcap` entry, then the data type defaults to ASCII.

With the ASCII data type the line printer spooler uses the `ln03rof` LN03R ASCII-to-PostScript translation filter. The `ln03rof` filter looks at the first two characters in the file and if the characters are %!, then the filter treats the file as a PostScript file and sends it to the printer without translation. If the characters are not %!, then the filter treats the file as an ASCII file and translates it before sending it to the printer.

The `ln03rof` filter sets a landscape print mode by default. To print using portrait mode, modify the `/etc/printcap` file to include the following two entries:

```
:pl#66:\
:pw#80:\
```

When printing in portrait mode, if the job page length is greater than 66 the page is printed in portrait, but a smaller font is used. If the job page width is greater than 80, the page is printed in landscape with a smaller font.

For more information about the `ln03rof` filter, see the `ln03rof`(8) reference page.

3.8.6.2 Problems Printing over TCP/IP Network – If you cannot print from your PrintServer client to the PrintServer printer over a TCP/IP network, type the following command to check the problem:

`% /etc/ping` *<PrintServer Node>*

If a problem does not exist, the TCP/IP PrintServer displays the following message:

<PrintServer Node> `is alive`

If a problem does exist, the TCP/IP PrintServer displays the following message (in approximately 30 seconds):

```
No answer from <PrintServer Node>
```

This message is displayed when your ULTRIX system has a problem resolving the TCP/IP network address (ARP). This is due to a deficiency in the TCP/IP PrintServer 20 that does not respond to the ARP broadcast.

To correct this problem, you must set the ARP entry manually by typing the following command in superuser mode on your ULTRIX system:

```
# /etc/arp -s <PrintServer Name> <Ethernet Address>
```

where:

<*PrintServer Name*> is the TCP/IP node name of the PrintServer 20.

<*Ethernet Address*> is the Ethernet address of the PrintServer 20.

Your PrintServer administrator can obtain the Ethernet address by typing the following command on the TCP/IP PrintServer's Supporting Host and executing the "configuration" option:

```
# lprc <PrintServer Name>
```

This example shows the format of the Ethernet address:

```
xx:xx:xx:xx:xx:xx
```

where: `xx` are the hexadecimal digits.

You must add the ARP entry to your `/etc/rc.local` file to ensure that this problem does not reappear after you reboot your ULTRIX system.

3.8.6.3 Job Fails with PostScript Error – If the job flag page prints, but the job itself does not print, check the device control library `Dl` entry in the `/etc/printcap` file matches the version of the PrintServer supporting host:

- If the PrintServer supporting host is Version 2.0 or 2.1, the `Dl` entry should be:

  ```
  :Dl=/usr/lib/lpdfilters/lps40.a:\
  ```

- If the PrintServer supporting host is Version 3.0, the `Dl` entry should be:

  ```
  :Dl=/usr/lib/lpdfilters/lps_v3.a:\
  ```

3.8.6.4 Sample Setup Modules for PrintServer – The *PrintServer Client for ULTRIX* describes controlling print jobs with printer setup modules by creating and sending the modules to the PrintServer system. It also shows examples for ANSI and PostScript setup modules.

The following are three examples of a PostScript setup module that you can use. These modules can be added to the `/usr/lib/lpdfilters/lps_v3.a` device control archive for your future use.

- The LPS_SEPARATE module:

```
%!
%! Module_name: LPS_SEPARATE
%! This module shows a sample setup module to begin each file
%! on a new sheet.  You can use this module to separate files
%! in a multi-file job.
%!
statusdict begin
newsheet
end
```

- The LPS_PS_EXT module:

```
%!
%! Module Name:  LPS_PS_EXT
%! This module shows how you can redefine POSTSCRIPT extensions
%! that are used by non-Digital printers to be Digital POSTSCRIPT
%! extensions. Note that for Control-D and Control-Z, you cannot
%! simply type ^D and ^Z. You must use the actual control codes
%! in the setup module.
%!
%! Please note that this example is based on North American
%! (letter) paper size. For Europe, change all the "lettertray"
%! references to "a4tray".
%!
%! Example:  Redefining POSTSCRIPT Extensions
%
/a3         {statusdict begin a3tray     end} def
/a4         {statusdict begin a4tray     end} def
/a5         {statusdict begin a5tray     end} def
/b4         {statusdict begin b4tray     end} def
/b5         {statusdict begin b5tray     end} def
/11x17      {statusdict begin 11x17tray  end} def
/ledger     {statusdict begin ledgertray end} def
/legal      {statusdict begin legaltray  end} def
/letter     {statusdict begin lettertray end} def
/note       {statusdict begin lettertray end} def
/statement {statusdict begin statementtray end} def
statusdict begin /statementtray /lettertray load def end
<03> cvn { } def % control-c
<04> cvn { } def % control-d
<14> cvn { } def % control-t
<1A> cvn { } def % control-z
```

- The LPS_PUNCHED module.

  This shows how you can print on either one or both sides of a sheet no matter how the paper is loaded. For the PrintServer 20, you have to load hole-punched paper differently depending on whether you want to print on one or both sides of a sheet. However, if you include a LPS_PUNCHED setup module in your print job, you can print on either one or both sides of a sheet no matter how the paper is loaded.

```
%!
%! Module Name: LPS_PUNCHED
%! This module shows how you can print on either one or both sides of a
%! sheet no matter how the paper is loaded.
%!
%!Example:  Printing on Punched Paper
%!
% Fix-Punched-Paper
%
% Use this module to invert the pages in a job printed on punched
```

```
% paper that is loaded the wrong way for the value of
% /Parameter=Sides that you are using.
/Fix-Punched-Paper-dict 20 dict def
Fix-Punched-Paper-dict begin

%
% Load the old values of the operators that we're going to redefine.
%

/old-showpage /showpage load def
/old-initgraphics /initgraphics load def
/old-initmatrix /initmatrix load def

%
% Determine the size of the paper. A dictionary is created that
% maps trays names into the size of the paper. Add other names
% if you are using other-sized paper.
%

5 dict begin
    /lettertray { 612 792 } def
    /a4tray { 595.28 841.89 } def
    currentdict                 % Leave this dict on stack.
    end

%
% Get the size of the current paper.
%

statusdict begin
    papersize pop               % Leave a name on stack.
    end

get exec                        % Look up the name in the dict.
/y-size exch def
/x-size exch def

%
% -- 'adjust-ctm' --
%
% Performs the inversion of the coordinate system.
%

/adjust-ctm {
    x-size y-size translate
    180 rotate
    } def

%
% This dictionary is used to hold our redefinitions of
% PostScript operators.
%

/redefinitions 3 dict def
redefinitions begin

%
% -- 'showpage' --
%
% Just like the old one, but invert the page before returning.
%

/showpage {
    Fix-Punched-Paper-dict begin
        old-showpage
```

```
        adjust-ctm
        end
    } def

%
% -- 'initgraphics' --
%
% Just like the old one, but invert the page before returning.
%

/initgraphics {
    Fix-Punched-Paper-dict begin
        old-initgraphics
        adjust-ctm
        end
    } def

%
% -- 'initmatrix' --
%
% Just like the old one, but invert the page before returning.

/initmatrix {
    Fix-Punched-Paper-dict begin
        old-initmatrix
        adjust-ctm
        end
        } def

end                                     % redefinitions

%
% Execute adjust-ctm to set things up for the first page.
%

adjust-ctm

%
% In order to get the "redefinitions" dictionary on the
% dictionary stack without our main dictionary, we leave
% it on the stack.

redefinitions

end                                     % Fix-Punched-Paper-dict

begin                                   % redefinitions.
userdict begin
```

3.8.6.5 Modification to lprsetup – The `lprsetup` program has been modified to provide a PrintServer queue entry in the `printcap` file. The following example shows the PrintServer queue entry that `lprsetup` produces:

```
lp0|lp|0:
     :ct=network:
     :lf=/usr/adm/lpd-errs/ex1:
     :of=lpscomm dotty %U %H %J:
     :ps=LPS:
     :sd=/usr/spool/lpd/ex1:
     :uv=4.0:
     :Da=ascii:
     :Dl=/usr/lib/lpdfilters/lps_v3.a:
     :Sd=a:
```

If you normally use European A4 size paper (8.3 x 11.7 inches) instead of American A size paper (8.5 x 11 inches), you should change the entry `:Sd=a:` to `:Sd=a4:.`

In the `lprsetup` program, if `lp` is set to `/dev/tty`*nn* you are prompted to choose between device or lat. The program is capable of checking whether or not it is a real lat device. If your printer is connected to a port on your machine, choose the default setting. If you want to access a printer by the lat, choose the lat option. The `lprsetup` program then checks if the device you have selected is a lat device. If it is, you are prompted to give information for the lat parameters, `ts`, `os` and `op`. You must set the `ts` parameter, and then either the `os` or the `op` parameter. If the tty*nn* is not a lat device, you receive a warning message.

3.8.6.6 New Entries in the printcap File – There are several new entries in the `printcap` file that are generated by the `lprsetup` program:

- `ct=<connection_type>`
- `uv=4.0`
- `ps=LPS`

3.8.6.7 ct=<connection_type> – The `:ct=<connection_type>:` entry determines the connection type. It is a new entry that has been added for ULTRIX/UWS Version 4.1 to provide a means of checking that parameters have been set up correctly. It is used to set any one of the following four connection types for a print queue:

- Device – The printer is connected to a serial or parallel port.
- LAT – The printer is connected to a LAT port.
- Remote – The job is submitted to a remote machine.
- Network – The job is submitted to a "foreign" print system by the network.

Each of these connection types provides a different way of connecting the output filter and the printer.

You need to set `ct` to specify the connection type you require in the `printcap` file for each print queue. The `ct` entry enables `lpd` to check that you have specified the correct parameters. If any of them have been omitted a diagnostic message is entered in the log file. However, you still have to specify the parameters you had to set to enable connections in earlier versions of the print system software. For example, the `rp` and `rm` parameters need to be set in the `printcap` file for each print queue where remote connections are allowed. See the `lpd`(8) reference page for a table showing which parameters should be used for which type of connection.

Note that only three of these connection types were available in earlier versions of the ULTRIX print system: device, LAT and remote. The fourth type of connection (network) has been added to support the LPS20 and LPS40 PrintServers in ULTRIX/UWS Version 4.1.

To enable PrintServer support you must ensure the `:ct=network:` entry is included in the `printcap` file for the appropriate print queue.

3.8.6.8 uv=4.0 – The `:uv=4.0:` entry determines the version number of the ULTRIX operating system. This entry has been introduced to allow backward compatibility with earlier versions of the print system. It is a replacement for the `:uv=psv1.0:` entry which was previously used in the `printcap` file to support PrintServer layered products.

If the `:uv=4.0:` entry is not included, then the print queue functions as it did in ULTRIX Version 3.1 or earlier and you cannot access the new features for PostScript support.

3.8.6.9 ps=LPS – The `:ps=LPS:` entry determines printer type. You must have this entry in the `printcap` file for each print queue using PrintServer features.

It allows `lpd` to assemble a PostScript job from the user's data files and PostScript device control modules. The device control modules access the device features and manipulate the appropriate printer parameters.

This entry also enables device control library support and provides an extensible mechanism for selecting translators to generate PostScript from `reGIS` or `ASCII` files.

3.8.6.10 Unknown Message from TCP/IP PrintServer – Sending a PostScript file containing a `returnstatus` PostScript operator to a TCP/IP PrintServer returns the following message:

```
Message number %XNNNNNNNN
```

where `NNNNNNNN` is an 8-digit hexadecimal number.

Because the `/usr/lib/lpdfilters/lps_v3.a` device control archive uses `returnstatus` for reporting errors you will receive a message number instead of an error message from the TCP/IP PrintServer. For example, attempting to print duplex on a PrintServer 40 which does not support duplex, returns a message number instead of the message "PrintServer 40 does not support duplex printing."

For example, if you get a message number,%X00000072, you can find out which error message it represents by consulting the following table:

Message Number	Explanation
Layup	
%X00000000	The current path may have been lost.
%X00000008	The 'copypage' is not supported by multipage layup.
%X00000012	Layup definition margins result in no usable sheet area.
%X0000001A	Layup definition margins overlap one another.
%X00000020	Pages per sheet are greater than number up. Pages per sheet set to number up.
%X00000028	First page is greater than pages per sheet. First page set to 1.
%X00000030	Tray selection is not supported by multipage layup.

Message Number	Explanation
LPS$$SetContext	
%X0000003A	No media-size medium is loaded in printer_name.
%X00000042	Media-size medium is not supported by printer_name.
%X0000004A	Media-size medium is not in the input_tray_name tray in printer_name.
%X00000054	Fatal device control library problem. Config error: configuration-error on printer_name.
%X0000005B	The upper page limit has been reached. Remaining pages will be flushed.
LPS$$SetOutputTray	
%X00000063	Output will be delivered to output-tray-name tray on printer_name.
%X0000006A	Tray-name tray on printer_name is tray-name.
LPS$$SetSides	
%X00000072	Printer-name does not support duplex printing.
%X0000007A	Printer-name does not support tumble printing.
%X00000082	Input tray selection not supported for printer_name.
%X0000008A	Output tray, output-tray-name, not supported on printer_name.
%X00000090	Condition on line line-number in layup definition.
Prologue Loader	
%X0000009B	Prologue prologue-name, version version-number.
%X000000A3	Bad password
%X000000AA	Duplex to 'face_up' output tray is not supported on printer_name.
%X000000B2	One_sided_duplex is not supported on printer_name.

For an explanation of error messages and user action, refer to the *ULTRIX Guide to Printserver Clients*.

3.8.6.11 ANSI Preamble Loading for TCP/IP PrintServer – The ANSI translator `ansi_ps` requires a preamble to be present in the printer. It is designed so that it sends this preamble to the printer unless it is invoked with an argument to indicate that the correct version of the preamble is already present.

The "VAX PrintServer Supporting Host Software" already supports loading the preamble at boot time, which improves the performance of short jobs by up to 9 seconds.

The current version of TCP/IP Supporting Host software does not implement the reporting of resources to `lpd`. As it is currently configured the ANSI translator always sends the preamble.

By using the following procedures you will enable the TCP/IP Supporting Host software to load the preamble when it boots, and disable the translator from sending the preamble.

To enable the TCP/IP Supporting Host software to load the preamble when it boots:

1. Log in to your TCP/IP Supporting Host machine and become the superuser.
2. Change to the configuration directory for the TCP/IP Supporting Host software and copy the ANSI preamble file from an ULTRIX/UWS Version 4.1 system:

   ```
   # cd /usr/lib/iplps
   # rcp <ULTRIX/UWS Version 4.1 node>:/usr/lib/lpdfilters/preamble.ps .
   ```

3. Add the following line at the top of the `preamble.ps` file:

   ```
   /decbind {} def
   ```

4. The modified `preamble.ps` file should now be appended to the file `setup.`<*printername*>.

   This command creates the file if it did not already exist:

   ```
   # cat preamble.ps >> setup.<printername>
   ```

5. Reboot your PrintServer.

To disable the translator from sending the preamble you should create a datatype called `fastansi.` You should propagate this change to every ULTRIX/UWS Version 4.1 machine from which you wish to send jobs to your PrintServer:

1. Edit the file `/usr/lib/lpdfilters/xlator_call`
2. Find the following lines:

   ```
   ansi)
        exec ansi_ps -F $pagesize -O $orientation -e "$@";;
   ```

3. Then add the following two extra lines immediately afterwards:

   ```
   fastansi)
        exec ansi_ps -F $pagesize -O $orientation
             -R "prologue lps_ansi_prologue Version 3.1-57" -e;;
   ```

You can now print jobs from your fast queue. For example,

```
% lpr -P <printername> -D fastansi file1 file2
```

3.8.7 PrintServer Layup Files Missing

The following four sample layup files are not located in `/usr/lib/lpdfilters` as stated in the *Guide to PrintServer Clients:*

- `lps`*n*`up.lup`
- `lpssingeholes.lup`
- `lpsdoubleholes.lup`
- `lpsholes.lup`

To work around the problem, you can create the files, then locate them in `/usr/lib/lpdfilters.` The files are data files and should be created with the owner as root, system as group, and permissions set to 644.

The contents of these files are as follows:

1. `lps2up.lup`

```
! LPS$2up.lup specifies a variation for 2 up printing.  A
! larger left margin is specified to allow for hole punching.
! This file is for single sided printing.

borders
margins = 19, 19, 60, 19
```

2. `lpsdoubleholes.lup`

```
! LPS$DoubleHoles.lup specifies a larger left margin to allow
! for hole punching.  This file is for double sided printing.

no borders
margins = 19, 19, 60, 19
alternate = left
```

3. `lpsholes.lup`

```
! LPS$holes.lup provides for a larger left margin for hole
! punching that will work in either duplex or simplex modes.
! Note that it will not produce the desired result with any
! tumbling operation.

no borders
margins = 19, 19, 60, 19
```

4. `lpssingleholes.lup`

```
! LPS$SingleHoles.lup specifies a larger left margin to allow
! for hole punching.  This file is for single sided printing.

no borders
margins = 19, 19, 60, 19
```

3.8.8 Configuring the System for an LA324 Printer

As there is no entry for the `la324` printer type in the `lprsetup` program, select the `lj250` printer type to add an LA324 print queue to your `/etc/printcap` file. When prompted for a symbol name, type `fs` and then change `03` to `023`. There are no other changes to make.

The following shows a sample entry for an LA324:

```
# This is for an LA324
lp5|5:\
        :af=/usr/adm/lp5acct:\
        :br#4800:\
        :ct=dev:\
        :fc#0177777:\
        :fs#023:\
        :if=/usr/lib/lpdfilters/lj250of:\
        :lf=/usr/adm/lp5err:\
        :lp=/dev/tty05:\
        :mc#20:\
        :mx#0:\
        :pl#66:\
        :pw#80:\
        :rw:\
        :sd=/usr/spool/lpd5:\
```

```
:sh:\
:uv=4.0:\
:xc#0177777:\
:xf=/usr/lib/lpdfilters/xf:\
:xs#044000:\
```

3.9 Software Development

This section contains notes about software development tools such as editors, compilers, libraries, linkers, and debuggers.

3.9.1 Customer Device Drivers: Recompile Potential

Customer device drivers might require recompiling and relinking if they use the kernel memory allocator interface defined in `/sys/h/kmalloc.h.`

Customer device drivers that use the memory allocator interface defined in `/sys/h/kmalloc.h` will no longer work if the macros KM_ALLOC, KM_FREE or KMEM_DUP are used. The allocator C routines `km_alloc`, `km_free` and `km_memdup` support the same formal arguments as in earlier versions of the operating system, and hence do not require changes to their usage. The macros have retained their original formal arguments; however, the macro implementation has changed.

A workaround for drivers that use the macros defined in `/sys/h/kmalloc.h` is to recompile dependent modules and then relink the kernel.

3.9.2 LANCE Driver Name Change

For the ULTRIX/UWS Version 4.1 release, the ULTRIX LANCE driver, formerly referred to as `se` has been renamed `ln`. Correspondingly, the `se`(4) manual page has been renamed to `ln`(4).

The LANCE driver is used on all ULTRIX workstations and servers which use the AMD LANCE Ethernet Controller chip as a network interface. Support exists for backward compatibility with previous releases. The system recognizes the `se` name and returns information for `ln` instead. However, any application which relies on the name `se` should be updated so that it does not depend on a specific hard coded name.

The most common case of hard coding an Ethernet device string into an application is to obtain the Ethernet hardware address. A portable method for obtaining this address (using sample code) is shown in the following example.

To obtain the Ethernet hardware address of the first available interface, follow an `ioctl` to get the interface configuration list with a call to get the interface flags for each interface. The flags' field is then checked for a valid interface, that is, one which is up and running. The actual name of the interface is not important. This allows future network devices to be recognized automatically, independent of the name assigned to them.

```
#define IFREQCNT 64

main() {
     ...
     struct ifreq *ifr;
     struct ifreq ifreqs[IFREQCNT];
```

```
        struct ifreq tmp_ifr;
        struct ifconf ifc;
        struct ifdevea ifrp;

        if((s = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
             perror("socket");
             exit( -1);
        }
        ifc.ifc_req = ifreqs;
        ifc.ifc_len = sizeof(ifreqs);
        if (ioctl(s, SIOCGIFCONF, &ifc) < 0) {
             perror("ioctl: SIOCGIFCONF");
             exit(-1);
        }

        /* loop through list of possible network interfaces */
        for(ifr = ifreqs; ifr < &ifreqs[IFREQCNT-1]; ifr++) {
             if(strlen(ifr->ifr_name) == 0) {
                  printf("No network devices configured\n");
                  exit(-1);
             }
             (void) strcpy(tmp_ifr.ifr_name, ifr->ifr_name);
             if (ioctl(s, SIOCGIFFLAGS, &tmp_ifr) < 0) {
                  perror("ioctl: SIOCGIFFLAGS");
                  exit(-1);
             }
             /* skip devices which aren't up and running, etc. */
             if (((tmp_ifr.ifr_flags & (IFF_UP|IFF_RUNNING)) !=
                       (IFF_UP|IFF_RUNNING)) ||
                  ((tmp_ifr.ifr_flags & (IFF_POINTOPOINT)) ==
                       (IFF_POINTOPOINT)) ||
                  ((tmp_ifr.ifr_flags & (IFF_DYNPROTO|IFF_BROADCAST)) !=
                       (IFF_DYNPROTO|IFF_BROADCAST))) {
                  continue; /* skip */
             }
             /* found a valid Ethernet interface */
             (void) strcpy(ifrp.ifr_name, tmp_ifr.ifr_name);
             break;
        }
        /* read the physical address */
        if(ioctl(s, SIOCRPHYSADDR, &ifrp) < 0) {
             perror("ioctl: SIOCRPHYSADDR");
             exit(-1);
        }
   }
```

3.9.3 BSD curses: Multiple Calls to initscr(), nocrmode() and nl() Cause Window Problems

It is documented that multiple calls to `initscr()` should not be issued. However, it would seem to be a safe assumption that if `initscr()` is called, then `endwin()` called before `initscr()` is called again, then the second call to `initscr()` should work correctly. A case has been discovered in which the second call does not work as expected.

The following example is extracted from a program which creates multiple subwindows and draws boxes around the windows. After initialization and establishment of "nocrmode" mode, several windows are created and drawn. One of the windows is activated, and the text "-- draw line no_crmode --" is displayed in it, followed by a second line asking the user to press the <RETURN> key. The windows are refreshed, closed (`endwin()`), then `initscr()` is called again,

followed by `crmode()`. This time the window edges are drawn to the left edge, instead of to the right edge.

```
#include      <curses.h>
#include      <ctype.h>
#include      <signal.h>
#include      <string.h>
      .
      .
      .

       WINDOW      *winb;

       extern      void      Die();
       extern      void      Window_open();
       extern      void      Initialize();
       extern      void      Hit_anykey();

void
main()
{
       initscr();
       nocrmode();
       nl();
       noecho();

       signal( SIGINT, Die );

       Window_open();
       Initialize();

       wmove( winb, 1, 5 );
       wprintw( winb, "-- draw line no_crmode --0" );
       wmove( winb, 2, 5);
       wprintw( winb, "-- Hit return key --0" );
       wrefresh( winb );
       Hit_anykey();
       endwin();

       initscr();
       crmode();
       nl();
       noecho();

       signal( SIGINT, Die );

       Window_open();
       Initialize();

       wmove( winb, 1, 5 );
       wprintw( winb, "-- draw line crmode --0" );
       wmove( winb, 2, 5);
       wprintw( winb, "-- Hit return key --0" );
       wrefresh( winb );
       Hit_anykey();
       endwin();

       Die();
}
```

The workaround is to reverse the order of the calls to `crmode()` and to `nl()` after the second call to `initscr()`, as follows:

```
initscr();
```

```
nl();
crmode();
noecho();
```

It is suspected that the `endwin()` function does not restore all terminal modes set prior to the first call to `initscr()`. It is not recommended, however, that `initscr()` be called more than once in a program.

3.9.4 Floating Point Emulation (RISC Only)

On RISC systems, if a floating point number is converted to an unsigned long, the value is truncated to the maximum positive signed long.

3.9.5 VAX pcc Compiler

The compiler does not convert unsigned integers to floating point correctly if the value exceeds the maximum positive signed integer.

3.9.6 RISC C Compiler

The following notes apply to the RISC C compiler:

- The RISC C compiler does not optimize the following program correctly:

  ```
  main()
  {
  int x,y;
  unsigned z;
  if (x-y+1 > z) good(); else bad();
  }
  ```

  The +1 gets lost. Use the `–O0` compiler option to disable all optimization, or cast z to `int` in the expression. Code that compares signed and unsigned characters in this fashion is problematical and should be avoided.

- The C compiler does not accept a constant pointer expression in a case statement. For example:

  ```
  (int)(&((REC *)0)->field)
  ```

  Remove the redundant cast, and it is accepted.

- When the `/tmp` file system is full, `cc` reports:

  ```
  ugen: internal  : line 0 : build.p,
  line 1743 unexpected u-code
  ```

3.9.7 RISC Program Size Defaults

The default maximum data and stack segment size for applications running on RISC systems has been changed to 32 Mbytes. Previous versions of RISC systems had an artificially high default of 88 Mbytes.

Data and stack segments can now be configured independent of swap space. To change the default, refer to Changes to Swap Space and Program Size Parameters in Chapter 1.

As a result of the change, some applications may not be able to extend their data. System calls will return the errno ENOMEM to these applications.

3.10 ULTRIX/SQL

The following notes apply to ULTRIX/SQL.

3.10.1 ULTRIX/SQL Commands Dump Core When the II_SYSTEM Variable Is Not Defined

As noted in several places in the ULTRIX/SQL documentation, the `II_SYSTEM` environment variable must be set for each ULTRIX/SQL user. If users enter ULTRIX/SQL commands when this variable is not defined, some of those commands will dump core without clearly reporting the error condition.

For more information about defining the `II_SYSTEM` variable, refer to the section that discusses startup files in either the *ULTRIX/SQL Installation Guide* or the *ULTRIX/SQL Release Notes*.

3.10.2 Layered Products Compatible with ULTRIX/SQL Version 1.1

Layered Products that run on ULTRIX/SQL Version 1.0 are compatible with ULTRIX/SQL Version 1.1. When either the ULTRIX/SQL Relational Database subset or the Development Library subset is loaded (QLRBASE110 or QLRDEVT110 for RISC; QLVBASE110 or QLVDEVT110 for VAX), two lock files are created, one for ULTRIX/SQL Version 1.1 and one for ULTRIX/SQL Version 1.0. This second lock file allows ULTRIX/SQL layered products which depend on the Relational Database or Development Library subset to load properly.

Note

If you are installing both ULTRIX/SQL and ULTRIX/SQL layered products from a RIS server environment, you must follow these steps:

1. Install the necessary ULTRIX/SQL subsets.
2. After the ULTRIX/SQL subsets have successfully installed, install the ULTRIX/SQL layered product subsets.

If you attempt to install the ULTRIX/SQL subsets and the ULTRIX/SQL layered product subsets simultaneously, the ULTRIX/SQL layered product subsets will fail to install.

3.10.3 New ULTRIX System Parameters Provided to Tune Priority Handling

In some circumstances, especially when there are a large number of database user processes, the normal reassignments of process priorities that accompany signal handling will have a detrimental effect on overall system performance. If this situation arises, it may be useful to reconfigure the `priority_mod` parameters provided at the end of the `/usr/sys/conf/mips/param.c` file for RISC systems or the `/usr/sys/conf/vax/param.c` file for VAX systems. The normal settings for these parameters are:

```
int sigpause_priority_mod = 0;
.
.
.
int psignal_priority_mod = 1;
```

To reconfigure the system so that process priority is not modified by signal activity, reset these parameters as follows:

```
int sigpause_priority_mod = 1;
.
.
.
int psignal_priority_mod = 0;
```

After you have reset these parameters, you must reconfigure and rebuild the kernel. For information on reconfiguring and rebuilding the kernel, see the *Guide to Configuration File Maintenance*.

3.10.4 ULTRIX/SQL rc.local Startup File Includes Multi-User Reentry Fix

In ULTRIX/UWS Version 4.0, ULTRIX/SQL provided the file `/usr/kits/sql/sqlstartup.rclocal` for you to include in your `/etc/rc.local` file to enable the automatic startup of SQL at system boot time. However, in that version, this file failed to restart the DBMS servers properly after the system was taken down to single-user mode and then brought back up to multi-user mode without being rebooted. This problem has been corrected. If you are updating your system from ULTRIX/UWS Version 4.0 to ULTRIX/UWS Version 4.1 you might want to include the new version of the ULTRIX/SQL startup file in `/etc/rc.local`.

3.10.5 ULTRIX/SQL Error Log File May Grow Very Large

The ULTRIX/SQL system maintains an error log in the file `/usr/var/kits/sql/sql/files/errlog.log`. This file can grow without bounds. It should be checked periodically and can be removed or truncated if previous messages logged there are no longer needed. If you remove the file, it will be re-created when new logging information is generated.

3.11 VAX C

The following notes apply to VAX C.

3.11.1 VAX C/ULTRIX (vcc) and pcc Calling Conventions

When calling a function that returns a structure, the VAX C for ULTRIX (vcc) compiler and the `pcc` compiler use incompatible calling conventions; this is the only case where the calling conventions differ. Specifically, if you call a function that returns a structure from `vcc` and that function was compiled with `pcc`, the call returns unpredictable results. If you call a function that returns a structure from `pcc` and that function was compiled with `vcc`, a segmentation fault occurs.

3.11.2 VAX C/ULTRIX (vcc) Compiler

When you use the quoted form of file inclusion, the `vcc` compiler looks for the included file in the directory the `vcc` command is executed from, not in the directory that contains the source file.

ULTRIX Worksystem Software Notes 4

This chapter discusses issues and known problems with the software and, when possible, provides solutions or workarounds to the problems.

The notes in this chapter cover the following topics:

- The X Windows System
- The Display Postscript System
- Fonts
- The User Environment

4.1 X Window System

This section contains notes pertaining to all ULTRIX/UWS Version 4.1 X servers, Xlib functions, and other X-related issues. For additional notes on X servers specific to individual processors, see Chapter 2, Processor-Specific Notes as well.

4.1.1 ULTRIX/UWS Version 4.1 X Servers

All the ULTRIX/UWS Version 4.1 X servers, except the `Xgb` server, have the Display PostScript System extension.

The ULTRIX/UWS Version 4.1 software contains the following X servers:

- `Xqdsg`: for 8- and 4-plane VAX color workstations
- `Xqvsm`: for VAX monochrome workstations
- `Xgb`: for VAXstation 3520/3540 workstations
- `Xcfb`: for DECstation/DECsystem 3100/2100, and DECstation/DECsystem 5000 Model 200 color workstations
- `Xmfb`: for monochrome DECstation/DECsystem 3100/2100 workstations
- `Xtm2d`: for DECstation/DECsystem 5000 Model 200PX workstations
- `Xtm`: for DECstation/DECsystem 5000 Model 200PXG and PXG-turbo workstations

All servers have been compiled under X11 Release 3.

4.1.1.1 Server-Client Interaction and DECnet Addressing – Starting with ULTRIX/UWS Version 4.0, DECnet addresses are specified with a variable length. The `dn_naddr` structure holds the address length in the first two bytes and the address itself in the remaining bytes.

In earlier releases of ULTRIX/UWS, DECnet addresses were two bytes long. The `dn_naddr` structure was always four bytes long; the first two bytes contained the length of the address, and the second two bytes contained the address. X servers that expect DECnet addresses to be fixed at two bytes in length will not communicate properly with some UWS X clients under ULTRIX/UWS Version 4.1.

Note that DECnet Phase V specifies that the address within `dn_naddr` is a variable-length address. DECnet code within the DECwindows Session Manager reserves space for DECnet Phase V addresses in the X protocol. This ensures compatibility with DECnet Phase V. To use the DECwindows Session Manager with a third-party (non-Digital) server requires that the server interpret `dn_naddr` as a variable-length structure.

For more information on `dn_naddr` and the X Protocol, see *X Window System: The Complete Reference to Xlib, X Protocol, ICCCM, XLFD*, Second Edition, Scheifler, Robert W. and James Gettys. Pages 463 and 734.

4.1.1.2 Default Keyboard Keymap – The US LK201-LA keyboard keymap is the default. To load a different keyboard keymap, you must first log in as superuser and then create a `keymap_default` symbolic link in the `/usr/lib/X11` directory, which points to the keyboard keymap you want to load. The following example shows you how to set the default keyboard keymap to the Swedish LK201:

```
# cd /usr/lib/X11
# ln -s keymaps/swedish_lk201lm.decw_keymap keymap_default
```

You must restart the X server after changing the default keyboard keymap. To restart the X server, type the following at the superuser prompt:

```
# /etc/shutdown now
# <CTRL-D>
```

To set the default keyboard keymap to US LK201-LA, you must remove the `/usr/lib/X11/keymap_default` file.

If you have multiple diskless clients (for example, three VAXstation 2000 systems) with different keyboards (German, French, English, Spanish) you cannot map the `keymap_default` entry of `/usr/lib/X11` to be private for each client.

4.1.1.3 Save-Unders and Backing Store – All X servers except the `Xgb` server now support save-unders as well as backing store functionality. Save-unders and backing store beautify screen and window refreshing, but increase the execution time of some windowing applications. Try running your server both with and without save-unders to decide if a significant performance penalty exists when using save-unders in your hardware and software configuration.

By default, save-unders and backing store are disabled on VAX systems and enabled on RISC systems. The state (enabled or disabled) of save-unders and backing store is determined by a command line option to the server executable. The command to invoke this executable is usually located in the `/etc/ttys` file. To turn save-unders and backing store on or off, you must edit the line in the `/etc/ttys` file where the server is invoked. The server must be restarted for any changes made in `/etc/ttys` to take effect.

To enable save-unders and backing store, invoke the server *without* the `-su` (save-under) and `-bs` (backing store) command line options. For example, to enable save-unders and backing store in the `Xcfb` color server, invoke the server with the following command:

```
# Xcfb
```

Save-unders can be disabled with the `-su` command line option. For example, to disable save-unders in the `Xcfb` color server, invoke the server with the following command:

```
# Xcfb -su
```

To disable both backing store and save-unders, invoke the server with the following command:

```
# Xcfb -su -bs
```

The `-bs` and `-su` server command line options are not yet documented in the reference pages.

4.1.1.4 Problems to Due Swap Space Size – If the system's swap space disk partition fills, an attempt by the server to access a previously unused page in the dynamically allocated portion of its data segment crashes the server with a segmentation fault. This problem can occur at various points during server execution, not just at those related to the allocation of server resources.

The anomaly stems from a peculiarity of ULTRIX virtual memory: Swap space on the disk partition is allocated when the memory is first accessed, and not when the corresponding memory segment is allocated.

To avoid the problem, configure your system with a larger swap space. Swap space requirements of the applications being used dictate the minimum acceptable swap partition size.

4.1.1.5 Invalid Font Path – If an invalid font path is specified on the ULTRIX server startup command line, the server will crash when it connects with the first client.

4.1.1.6 Host Names in X Server Access Control List – When the X server starts up, it places the name of the local host and the name of the host where the X server is running into the server's access control list. In addition, if there is a `/etc/X0.hosts` file or `/etc/X1.hosts` file, its contents are added to the access control list.

If you have `sm.host_list` resource in your `.Xdefaults` file, the access control list replaces the server's list when `dxsession` starts up. If no such resource exists, `dxsession` will not change the server's list.

When you display the Session Manager Customize Security dialog box, the server's current access control list is listed. If this list is changed and the current settings are saved, a `sm.host_list` resource is placed into the `.Xdefaults` file. The next time you start `dxsession`, the server's list will be replaced by this resource.

4.1.1.7 X Server Messages File –

X server messages are logged in the `/usr/adm/X#msgs` file. The file contains the date and time the X server restarts, the X error messages, and miscellaneous information about X server crashes at server restart. The number sign (#) indicates the number of the display where messages are being logged. For workstations with single displays, the messages file name is `X0msgs`. For workstations with two displays (such as the VAXstation II/GPX) there are two messages files, `X0msgs` and `X1msgs`, the first and second display respectively.

Messages reported to the `X#msgs` file include:

- "Use" messages, which indicate that the line in the `/etc/ttys` file for starting the server has a typographical error. Check the `/etc/ttys` file for spelling or syntax errors.
- "Resource missing" or "Resource installed improperly" messages, which indicate that fonts are not installed properly, or that the `-fp` switch in the `/etc/ttys` line is not correct. Check to see that the fonts you are using are installed properly, that the `-fc` option for setting the default cursor font is correct, and that the `-fn` option for setting the default text font is correct.

  For example:

  ```
  main: Could not open default font 'XXX'

  main: Could not open default cursor font 'XXX'
  ```

- "Could not open RGB_DB 'XXX'" messages, which indicate that the rgb database is not installed correctly. These messages refer to files that apply to color workstations only. For VAXstation 3520/3540s, the database files are in `/usr/lib/rgb.*`. For all other color workstations, the database files are in `/usr/lib/X11/rgb.*`
- "Screen failed initialization" messages, which usually indicate that the file `/dev/mouse` is not protected correctly or there is already a server running.
- Other error messages that indicate that the X server is out of memory.
- Some of the X messages are fatal errors.

4.1.1.8 How to Restart the X Server –

The following procedure briefly describes the simplest way to restart the X server. Use this procedure if your X server hangs:

1. Log in to your workstation from a remote terminal.
2. Change your user ID to `root` using the `su`(1) command.
3. Enter the following command:

   ```
   # ps -ax
   ```

4. Locate the X server process. The output of the `ps`(1) command should display an X server process similar to the following:

   ```
   PID TT STAT  TIME COMMAND
    95 ?  S   117:09 - :0 (Xqdsg)
   ```

5. Kill the X server PID by entering the following command:

   ```
   # kill -9 95
   ```

By killing the X server process, the server restarts assuming that the server startup line is the file `/etc/ttys`.

4.1.1.9 LockDisplay and UnLockDisplay Macros – The X Window System programming manuals mention use of `LockDisplay` and `UnlockDisplay` macros for writing multithreaded X clients that access buffers of the Display structure. These macros are often used when writing Xlib side extensions to the core X11 protocol.

However, ULTRIX/UWS Version 4.0 and ULTRIX/UWS Version 4.1 do not support these side extensions. Hence, the `LockDisplay` and `UnlockDisplay` macros are not included. If you require these macros, order the ULTRIX/UWS Version 4.1 source kit. Note that you may be able to accomplish display structure locking using the public domain macros provided by MIT's Xlib.

Note also that supported core Xlib calls do proper structure locking and are safe for use by multithreaded clients.

4.1.1.10 Memory Allocation Routines – Xlib defines its own internal versions of the `malloc()`, `calloc()`, `free()` and `realloc()` routines, which are also used by the XUI Toolkit. All clients that link with Xlib or the XUI Toolkit should use the Xlib-defined versions of these routines. The `malloc()`, `free()`, `realloc()`, and `calloc()` routines are defined in `XvmsAlloc.o` in `libX11.a`.

If you prefer to use your own version of one or more of these routines, then you must take action to avoid multiple declaration errors when linking. You can do this by declaring entry points in your code for each of these four routines and linking normally with this command:

```
% cc prog1.c -lX11
```

Alternatively, note that in `/usr/lib/libc.a`, `calloc()` is defined in `calloc.o`, but `malloc()`,`realloc()`, and `free()` are defined in `malloc.o`. Therefore, if you declare your own routines for `malloc()`, `realloc()`, and `free()`, but want to pull in `calloc()` from `/usr/lib/libc.a`, you can do so with these commands:

```
% ar x /usr/lib/libc.a calloc.o
% cc prog2.c calloc.o -lX11
```

Some popular public domain software packages define their own versions of some of these memory allocation functions. You can compile public domain programs that define their own memory allocation functions under ULTRIX/UWS Version 4.1 by doing one of the following:

- Disable the package's version of the functions. They typically have an option or flag you can set to request that the application use the system memory allocation functions.
- Add function definitions for the functions not defined. Typically, `calloc` is missing. The `calloc` from `libc`, as described above, may be added.
- Disable the functions in Xlib by removing the `XvmsAlloc.o` module from `libX11.a`.

4.1.1.11 ULTRIX System V Emulation Library – To use the ULTRIX System V emulation library you must link `/usr/lib/XlibIntV.o` before `/usr/lib/libX11.a`. For example, you would enter this command:

```
% cc xsample.c /usr/lib/XlibIntV.o -lX11 -lcV
```

4.1.1.12 XCopyArea Function – An application may hang and freeze up the system with an `XCopyArea` function call followed by an `XIfEvent` call. To avoid the possibility of deadlock, use an `XSync` function call before issuing the `XIfEvent` function call. The `XSync` call will flush the output buffer and wait for all requests to be received and processed by the server. In general, if `XIfEvent` is waiting for an expected event generated by an `X` function call, `XSync` should first be used to guarantee that all requests to the server have been sent out. The application must guarantee that all events are truly written out before the wait is begun.

4.1.1.13 XDrawArc(s) Function – Double dash mode does not work with the `XDrawArc`(s) function. The `XDrawArc` function is used to draw circles and ellipses. The `XDrawArc` function is used by some MIT applications.

4.1.1.14 Data Structures and Constants – In UTLRIX/UWS Version 2.2, additions were made to the `XSizeHints` and `XStandardColormap` data structures and a new structure was added for manipulating properties containing text.

4.1.1.14.1 X Size Hints – The following routines are used for getting and setting size hints:

The `XGetWMSizeHints` routine returns the size hints stored in the indicated property on the specified window. If the property is of type WM_SIZE_HINTS, of format 32, and is long enough to contain a size hints structure, the various fields of the `hints_return` structure are set and a nonzero status is returned. Otherwise, a status of 0 is returned. To get a window's normal size hints, the `XGetWMNormalHints` routine may be used instead.

```
Status XGetWMSizeHints (dpy, w, hints_return, property)
    Display *dpy;
    Window w;
    XSizeHints *hints_return;
    Atom property;
```

The `XGetWMSizeHints` routine can generate BadWindow, BadAtom, or BadValue errors.

The `XGetWMSizeHints` routine supersedes `XGetSizeHints`.

The `XSetWMSizeHints` routine replaces (or sets if the property does not exist) the size hints for indicated property on the specified window. The property is stored with a type of WM_SIZE_HINTS and a format of 32. To set a window's normal size hints, the `XSetWMNormalHints` routine may be used instead.

```
void XSetWMSizeHints (dpy, w, hints, property)
    Display *dpy;
    Window w;
    XSizeHints *hints;
    Atom property;
```

The `XSetWMSizeHints` routine can generate BadWindow, BadAtom, or BadValue errors.

The XSetWMSizeHints routine supersedes XSetSizeHints.

The XGetWMNormalHints routine returns the size hints stored in the WM_NORMAL_HINTS property on the specified window. If the property is of type WM_SIZE_HINTS, of format 32, and is long enough to contain a size hints structure, the various fields of the hints_return structure are set and a nonzero status is returned. Otherwise, a status of 0 is returned.

```
Status XGetWMNormalHints (dpy, w, hints_return)
    Display *dpy;
    Window w;
    XSizeHints *hints_return;
```

The XGetWMNormalHints routine can generate BadWindow, BadAtom, or BadValue errors.

The XGetWMNormalHints routine supersedes XGetNormalHints.

The XSetWMNormalHints routine replaces (or sets if the property does not exist) the size hints for the WM_NORMAL_HINTS property on the specified window. The property is stored with a type of WM_SIZE_HINTS and a format of 32.

```
void XSetWMNormalHints (dpy, w, hints)
    Display *dpy;
    Window w;
    XSizeHints *hints;
```

The XSetWMNormalHints routine can generate BadWindow, BadAtom, or BadValue errors.

4.1.1.14.2 XStandardColormap – Two new elements were added in ULTRIX/UWS Version 2.2 to properties of type RGB_COLOR_MAP: the id of the visual from which the colormap was created, and an arbitrary resource id that indicates whether or not the cells held by this standard colormap should be released by freeing the colormap id or by doing a KillClient on the indicated resource (necessary for allocating out of an existing map). These fields can be added to the end of the existing structure (defined in Xutil.h) without disrupting the existing interfaces:

```
typedef struct {
        Colormap colormap;
        unsigned long red_max;
        unsigned long red_mult;
        unsigned long green_max;
        unsigned long green_mult;
        unsigned long blue_max;
        unsigned long blue_mult;
        unsigned long base_pixel;
        VisualID visualid;       /* NEW */
     XID killid;          /* NEW */
} XStandardColormap;
```

In addition, a new symbolic value was added to Xutil.h indicating resources have been released by freeing the colormap:

```
#define ReleaseByFreeingColormap ((XID) 1L)
```

4.1.1.14.3 XTextProperty – Many of the properties of type STRING were changed in ULTRIX/UWS Version 2.2 to allow a variety of types and formats. Since the data stored in these properties are no longer simple null-terminated strings, a new data structure describes the encoding, type, length, and value of the text as well as its value. The following structure was added to `Xutil.h`:

```
typedef struct {
    unsigned char *value;    /* property data */
    Atom encoding;           /* type of property */
    int format;              /* 8, 16, or 32 */
    unsigned long nitems;    /* number of items in value */
} XTextProperty;
```

4.1.1.14.4 WithdrawnState Constant – Even though interfaces to the WM_STATE property are not being defined, the constant `WithdrawnState` was added to `Xutil.h` with a value of zero.

4.2 Display PostScript System

The Display PostScript System from Adobe Systems, Inc., extends the entire PostScript language to bitmap displays. UWS implements the Display PostScript System as an extension to the UWS server. Using this extension, DECwindows client applications can send both PostScript imaging requests and X requests to the same drawable, using a single network connection.

4.2.1 Example Programs Using the Display PostScript System

The `/usr/examples/dps` directory contains example programs that use the Display PostScript System. For more information about these example programs, see the *Guide to Developing Applications for the Display PostScript System*.

4.2.2 Additional Documentation

In addition to the documentation described in the *Guide to Developing Applications for the Display PostScript System*, developers can refer to Adobe's Document Structuring Conventions specification, available free of charge from Adobe Systems, Inc.,

To request a copy by electronic mail, send a mail message to the Adobe file server at either of the following network addresses:

Internet	`ps-file-server@adobe.com`
UUCP	`...!decwrl!adobe!ps-file-server`

4.2.3 Allocating a Colormap for Use with Display PostScript

The colormap cells allocated in a color cube or gray-scale ramp must be contiguous. The `XAllocColorCells` routine can be used to allocate contiguous planes, but not contiguous color cells. Using noncontiguous color cells produces undefined results.

The following routine allocates *n* contiguous colormap cells, assuming they exist in the specified colormap. Use this routine, rather than `XAllocColorCells`, when defining a colormap or gray-scale ramp for use with Display PostScript code.

```
static Bool
_AllocContiguousCells(dpy, cmap, pixels, npixels)
Display *dpy;
Colormap cmap;
unsigned long *pixels;          /* filled in by routine */
int npixels;
{
 unsigned long *waste = (unsigned long *)NULL;
    int nwaste = 0;
    Bool contig = False;
    int status, i;
    unsigned long masks = NULL;

    while (!contig) {
        status = XAllocColorCells(dpy, cmap, False, &masks, 0, pixels,
          npixels);
        if (!status)
            break;   /* can't get enough contiguous cells */
        for (i=0; i < (npixels-1); i++) {
            if (pixels[i] + 1 != pixels[i+1]) {
                /* isn't contiguous, keep trying */
                XFreeColors(dpy, cmap, &pixels[i+1], npixels - (i+1), 0);
                if (!waste)
              waste = (unsigned long *)malloc((i + 1) *
                                sizeof(unsigned long));
          else
                    waste = (unsigned long *)realloc(waste, (nwaste +
                                (i + 1)) * sizeof(unsigned long));
 bcopy(pixels, waste+nwaste, (i+1) * sizeof(unsigned long));
                nwaste += (i+1);
                break;
            }
        }
        if (i == (npixels-1))
            contig = True;
    }
    /* clean up and return 'contig' */
    if (nwaste) {
        XFreeColors(dpy, cmap, waste, nwaste, 0);
        free(waste);
    }
    return (contig);
}
```

4.2.4 setrgbXactual Operator Name Change

The `setrgbXactual` operator has been renamed to `setXrgbactual`, for consistency in naming with the other `X` specific operators. The name `setrgbXactual` remains as an alias.

4.2.5 Contexts Created Using the Default Colormaps

Display PostScript programs that use `XDPSCreateSimpleContext()` to create display PostScript contexts with the default colormaps have slightly inferior color rendition. Colors are chosen from a palette of 64 colors (eight greys) instead of 125 colors. Applications that need to use more than 64 colors can get them by using the `XDPSCreateContext()` context, which allows the use of an unlimited number of colormap cells.

4.2.6 Changing the Default XStandardColormap

If you want to alter the `XStandardColormap` default used by simple Display PostScript clients, or if you want to free the colormap cells used by the `XStandardColormap` routine you can do so by following the methods described in the ICCCM.

Note

Before you attempt to change the `XStandardColormap` default, you must be certain that no clients are using `XStandardColormap`.

4.2.7 Automatic PostScript Garbage Collection

Automatic PostScript garbage collection is turned on by default. It is a global setting to the server, but it is turned on every time `start` executes, that is, each time a client creates a context.

Any client may turn off automatic garbage collection, but doing so turns it off for all clients. Garbage Collection is turned back on the next time a client creates a context.

There is no way to determine the current state of automatic garbage collection.

4.3 Fonts

This section explains how fonts are installed, named, and organized in directories. It also provides notes for application programmers on using fonts.

Fonts are installed by UWS (or by individual applications) and are read by the X servers on behalf of applications. An X Consortium standard defines the bitmap distribution format (BDF) in which font sources are distributed. However, X servers and applications use fonts in a compiled format, which is not standardized.

Most of the ULTRIX/UWS Version 4.1 servers use fonts in the X11 portable compiled font (PCF) format. This represents a change from the ULTRIX/UWS Version 2.1 release, where compiled fonts were in DECwindows format (DWF) or server natural format (SNF). (The `Xgb` server for VAXstation 3520 and 3540 systems, however, continue to use DWF format.) ULTRIX/UWS Version 4.0 also made changes to the directories in which fonts are located and expanded the directory structure for user-supplied fonts and application-supplied fonts.

ULTRIX/UWS Version 4.0 also added fonts for use with the X implementation of Display PostScript (XDPS). These fonts have their own format, which is a PostScript-compatible ASCII format. If you have additional PostScript-compatible fonts, you can install them for use with Display PostScript.

If you are a system manager or applications programmer, you should familiarize yourself with how the font format change and new directories might affect your environment.

The subsections of this section are as follows:

- Fonts and font utilities on the kit
- Default font directories
- Application-specific and custom fonts

- Display PostScript fonts
- Application font information for developers
- Font names and aliases
- Layered applications with fonts
- Changes to the terminal font

4.3.1 Fonts and Font Utilities

The ULTRIX/UWS Version 4.1 software includes font sets for 75 dots-per-inch (dpi) and 100 dpi displays.

To use fonts other than those supplied in this kit, you must compile their `.bdf` font source files. Use the `/usr/bin/dxfc` font compiler to create `.pcf` files. (For the `Xgb` server, use `/usr/bin/dxfc3d` to create `.dwf` files.) Next, use `/usr/bin/dxmkfontdir` to create a list of the fonts in the directory for use by the X server. For more information, refer to `dxfc(1X)` and `dxmkfontdir(1X)` in the *UWS Reference Pages*.

ULTRIX/UWS Version 4.1 also contains 123 MIT X11 Release 3 fonts for 75 dpi displays and fonts for compatibility with X10 applications. The MIT and X10 compatibility fonts are unsupported, and are packaged in the unsupported portion of the software distribution.

4.3.2 Default Font Directories

To be usable, a font must be installed in a directory on the X server's font search path. The default fonts for ULTRIX/UWS Version 4.1 are the 75 dpi fonts, but you can install and use 100 dpi fonts.

The server's font search path does not include both 75 dpi and 100 dpi font directories. To use the 100 dpi fonts, you must alter the font path.

4.3.2.1 75 dpi Fonts – If you install just the mandatory subsets, you install compiled 75 dpi fonts (the default fonts). The server directory search path for these fonts is as follows:

```
/usr/lib/X11/fonts/decwin/75dpi/
/usr/lib/X11/fonts/MIT/
/usr/lib/X11/fonts/compX10/
/usr/lib/X11/fonts/local/75dpi/
/usr/lib/X11/fonts/apps/75dpi/
/usr/lib/X11/fonts/private/75dpi/
```

When a font is requested (by the Xlib routine `XQueryFont` for example), the DECwindows X server looks for the font first in `/usr/lib/X11/fonts/decwin/75dpi/`, then in `/usr/lib/X11/fonts/MIT/`, and so on.

The 75 dpi fonts for the `Xgb` server are in DWF format; they are installed in the following directories:

```
/usr/lib/dwf/75dpi
/usr/lib/dwf/compX10
/usr/lib/dwf/mitX11
```

4.3.2.2 100 dpi Fonts – The ULTRIX/UWS Version 4.1 installation procedure allows you to install 100 dpi fonts in an optional subset. The server search path for 100dpi fonts is as follows:

```
/usr/lib/X11/fonts/decwin/100dpi/
/usr/lib/X11/fonts/MIT/
/usr/lib/X11/fonts/compX10/
/usr/lib/X11/fonts/local/100dpi/
/usr/lib/X11/fonts/apps/100dpi/
/usr/lib/X11/fonts/private/100dpi/
```

The 100 dpi fonts for the `Xgb` server are installed in the following directories:

```
/usr/lib/dwf/100dpi
/usr/lib/dwf/compX10
/usr/lib/dwf/mitX11
```

To use the 100 dpi fonts, you must put them on the X server's search path by editing the workstation's `/etc/ttys` file. The server startup line looks like the following:

```
:0 "/usr/bin/login -P /usr/bin/Xprompter -C /usr/bin/dxsession" none on
secure window="/usr/bin/Xcfb"
```

Modify it to include the `-fd` and `-dpi 100` switches as follows:

```
:0 "/usr/bin/login -P /usr/bin/Xprompter -C /usr/bin/dxsession" none on
secure window="/usr/bin/Xcfb -fd 100 -dpi 100"
```

A ULTRIX/UWS Version 4.1 font search path uses only the option specified in the `/etc/ttys` file.

After editing the `/etc/ttys` file, restart the X server to change the default font search path.

4.3.2.3 Font Directory Contents – This section describes the contents of the font directories.

- `/usr/lib/X11/fonts/decwin/75dpi/` and `/usr/lib/X11/fonts/decwin/100dpi/`

  ULTRIX/UWS Version 4.1 uses these font directories for its supported fonts. Reserve these directories for use by the UWS software (consider them as being for *read only* use). These directories might be moved or deleted in future releases of UWS. These directories are shared in the diskless environment.

- `/usr/lib/X11/fonts/MIT/`

  This font directory holds fonts distributed by the MIT X Consortium. The MIT fonts are unsupported and in future releases of UWS the directory and its contents might be moved or deleted. Consider the directory as being for *read only* use. It is shared in the diskless environment. Applications should not install fonts in this directory.

- `/usr/lib/X11/fonts/compX10/`

  This directory contains unsupported fonts needed for compatibility with X10. The directory and its contents might be moved or deleted in future releases of UWS. Consider this directory for *read only* use. It is shared in the diskless environment. Applications should not install fonts in this directory.

- `/usr/lib/X11/fonts/local/75dpi/` and `/usr/lib/X11/fonts/local/100dpi/`

  These font directories are a local version of `.../fonts/decwin`. You can use these directories to install site-specific fonts, such as the corporate logo. The directories are unsupported, and no maintenance is performed during the installation. However, they are in the default server font path, so fonts placed here are automatically available. These directories are not intended for use by applications. They are shared in the diskless environment.

- `/usr/lib/X11/fonts/apps/75dpi/` and `/usr/lib/X11/fonts/apps/100dpi/`

  Shared applications can install fonts in these directories. Like the app-default file or class names, there is no registry; applications must use unique names to avoid collisions. The directories are unsupported, and no maintenance is performed during installation. However, they are in the default server font path, so fonts placed here are automatically available. They are shared in the diskless environment.

- `/usr/lib/X11/fonts/private/75dpi/` and `/usr/lib/X11/fonts/private/100dpi/`

  Applications that install fonts that are licensed on a per-workstation basis can install fonts in these directories. Like app-default files or class names, there is no registry; applications must use unique names to avoid collisions. These directories are unsupported, and no maintenance is performed during installation. These directories are in the default server font path. They are not shared in the diskless environment.

- `/usr/lib/DPS/outline/`

  This directory contains subdirectories for Display PostScript (DPS) fonts. The DPS font directories differ from the previously-described server font directories. The fonts are not `.pcf` files and they are not on the server search path; they are used by DPS only. The `decwin` subdirectory holds the default Display PostScript fonts installed by ULTRIX/UWS Version 4.1 The `local` subdirectory is where you can install site-specific fonts for DPS; this directory is shared in the diskless environment. The `apps` subdirectory is for applications that install fonts for sharing in the diskless environment. The `private` directory is where applications can install fonts that are not shared.

4.3.2.4 Installation Subsets and Server Font Directories – The following tables list the directories into which installation subsets install supported server fonts, unsupported server fonts, and Display PostScript fonts, respectively.

Subset	Font Location(s)
UDTAFM410	`/usr/lib/font/metrics`
ULTAFM410	`/usr/lib/font/metrics`
UDWFONT410	`/usr/lib/X11/fonts/decwin/75dpi`
UWSFONT410	`/usr/lib/X11/fonts/decwin/75dpi`
UDWFONT15410	`/usr/lib/X11/fonts/decwin/100dpi`

Subset	Font Location(s)
UWSFONT15410	/usr/lib/X11/fonts/decwin/100dpi
UWS3DFONT410	/usr/lib/dwf/75dpi, /usr/lib/dwf/100dpi

Subset	Font Location(s)
UDXUNCOMP410	/usr/lib/X11/fonts/compX10
ULXUNCOMP410	/usr/lib/X11/fonts/compX10
UDXUNMIT410	/usr/lib/X11/fonts/MIT
ULXUNMIT410	/usr/lib/X11/fonts/MIT
UWS3DFONT410	/usr/lib/dwf/mitX11, /usr/lib/dwf/compX10

Subset	Font Location(s)
UDWSER410	/usr/lib/DPS/outline/decwin
UWSSER410	/usr/lib/DPS/outline/decwin

4.3.3 Application-Specific and Custom Fonts

The X servers can read fonts in PCF format, compressed PCF format, and BDF format (except for the `Xgb` server, which reads only DWF format fonts, as in a previous UWS release). It is best to compile fonts into PCF format. Compressed PCF files cause slightly slower performance; the uncompiled BDF files substantially degrade the server's performance.

The servers do not read:

- Fonts produced by the MIT X11 Release 3 compiler (whose file suffix is
- Fonts produced by the DECstation/DECsystem 3100 ULTRIX/UWS Version 2.1 (RISC) font compiler (whose file suffix is `.snf`)
- Fonts produced by the VAX ULTRIX/UWS Version 2.1 font compiler (whose file suffix is `.dwf`).

To compile and install BDF fonts:

1. Compile the fonts using the `dxfc` command. (Use the `dxfc3d` command if you have a `Xgb` server.) Put the output in a directory that is on the default font path, such as `/usr/lib/X11/fonts/local/75dpi/`.
2. Use the `dxmkfontdir` utility to create a list of the fonts in the directory. Put the list into a file named `fonts.dir`.

Users can now access the fonts when they log in. To compress fonts into `.pcf.Z` files, use the `compress` utility.

To ensure that your fonts are always available in the X server's font search path, place fonts in the default font directories. If you choose to install your fonts elsewhere, alter the search path to make the fonts available by editing the `/etc/ttys` file for each workstation.

In the /etc/ttys file, add the new directory to the server startup line. Separate multiple font directories with commas. Use the +fp command line option and the following format to prepend elements to the font path:

```
+fp path[,path...]
```

Use the fp+ option and the following format to append elements to the font path:

```
fp+ path[,path...]
```

For example:

```
:0 "/usr/bin/login -P /usr/bin/Xprompter -C /usr/bin/dxsession" none on
secure window="/usr/bin/Xcfb fp+ /udir/susan/toyfonts/"
```

Note that if the /etc/ttys file is ever deleted, as it could be during a system software update, the X server will be unable to find these fonts until you again edit the file. For this reason, using the Digital-supplied font directories is the preferred practice (refer to Section 4.3.2.3 for descriptions of the font directories).

4.3.4 Display PostScript Fonts

By installing the ULTRIX/UWS Version 4.1 software, you obtain all of the fonts you need to use Display PostScript. These fonts are in the /usr/lib/DPS/outline/decwin directory. If you want to install additional PostScript-compatible ASCII fonts to use with Display PostScript, place them in the directories listed in Section 4.3.2. Restart the X server after installing the fonts.

4.3.5 Application Font Information for Developers

There is no need to recompile fonts when upgrading applications from ULTRIX/UWS Version 4.0 to ULTRIX/UWS Version 4.1. Fonts distributed with your application must be recompiled when upgrading from ULTRIX/UWS Version 2.1 or ULTRIX/UWS Version 2.2 to ULTRIX/UWS Version 4.0. Applications should ship fonts in BDF format and compile them as part of the installation. This is necessary because the format of compiled fonts is variable. For example, the VMS operating system and the ULTRIX operating system have different formats for compiled fonts and, as noted previously, ULTRIX/UWS Version 4.0 and ULTRIX/UWS Version 4.1 use a format that differs from that of the previous version of UWS. Compiling fonts as part of the installation protects your application from future format changes. For information on compiling and installing fonts, refer to Section 4.3.3.

4.3.6 Font Names and Aliases

Font names consist of a series of parameter values separated by dashes, describing the typographic characteristics of the font. However, by using aliases, you can use a set of less cumbersome names.

4.3.6.1 Font Names – ULTRIX/UWS Version 4.1 uses the naming convention specified by the standard *X Logical Font Description Conventions, X Window System, Version 11*. UWS font names are specified using the logical font descriptions for the X protocol (XLFDs). A sample font name is as follows:

```
-ADOBE-ITC Avant Garde Gothic-Book-R-Normal--14-100-100-100-P-80-ISO8859-1
```

In left-to-right order, these are the fields in a font name and the values in the sample name:

Field	Sample Value
Foundry	Adobe
Family_Name	ITC Avant Garde Gothic
Weight_Name	Book
Slant (Roman (R), Italic (I), Oblique (O), Reverse Italic (RI), Reverse Oblique (RO) or Other (OT))	R
Setwidth_Name	Normal
Pixel_Size	14
Point_Size, in decipoints	100
Resolution, horizontal and vertical, in pixels/dots per inch	100 dots per inch (horizontal) 100 dots per inch (vertical)
Spacing (Proportional (P), Monospaced (M), CharCell (C))	P
Average_Width, in decipoints	80
Charset_Registry	ISO8859
Charset_Encoding	1

A comparable font for a 75 dpi screen has the following name:

```
-ADOBE-ITC Avant Garde Gothic-Book-R-Normal--10-100-75-75-P-59-ISO8859-1
```

This font needs 10 pixels to appear as 10 points. This font differs from the previous sample font only in pixel size, resolution, and character width.

You can use wildcards in font names to specify the values of certain font characteristics and allow the server to provide the appropriate values for those fields that differ on different workstation screens. A question mark (?) wildcard substitutes for a single character, and an asterisk (*) wildcard substitutes for one or more fields. The following font name specifies a 10-point ITC Avant Garde Gothic font of book weight, roman style, and normal spacing for display on either 75 or 100 dpi systems:

```
-ADOBE-ITC Avant Garde Gothic-Book-R-Normal--*-100-*-*-P-*
```

When you use the asterisk, be sure that the substitutions resolve clearly. For example, in the following font name, the leftmost asterisk substitutes for two fields before the 100:

```
-ADOBE-ITC Avant Garde Gothic-Book-R-Normal--*-100-*-P-*
```

The name resolves to two fonts:

```
-ADOBE-ITC Avant Garde Gothic-Book-R-Normal--11-80-100-100-P-80-ISO8859-1
-ADOBE-ITC Avant Garde Gothic-Book-R-Normal--14-100-100-100-P-80-ISO8859-1
```

The first is an 8-point font; the second is a 10-point font. The server chooses one of the fonts. ULTRIX/UWS Version 4.1 servers choose the first font (in ASCII-sorted order), but applications should avoid dependence on this algorithm, as it could

change in a future release. For more on specifying fonts from applications, refer to the "Application Font Information for Developers" section.

4.3.6.2 Specifying Fonts – An application should use wildcards for display-specific fields in a font name. When the wildcards in the name are resolved, the resulting font will match both the application's needs and the characteristics of the workstation display.

Resolution and pixel size are characteristics that vary on different displays, and these are good properties for which to use wildcards. For example, an application can specify the FAMILY_NAME, WEIGHT_NAME, and POINT_SIZE properties for a font, omitting RESOLUTION_X, RESOLUTION_Y, and PIXEL_SIZE. This produces the correct physical font size for the workstation's display resolution at startup.

Applications reading font specifications from a defaults file should use `XListFonts` or `XListFontsWithInfo` to query the server for a list of matching fonts. The application itself could then resolve the wildcards before requesting an open font. In this way, the application controls the font selection method.

If the application instead simply passes the font name with wildcards to the server in the argument to an `OpenFont` request, the server resolves the wildcards. The server has a simple selection method that might not produce the font most beneficial to the application. The algorithm for choosing a font from an ambiguous font name pattern is server-dependent.

4.3.6.3 Font Name Aliases – ULTRIX/UWS Version 4.1 font files have lowercase names that indicate the contents of the file, and they have a `.pcf` suffix (except for fonts for the `Xgb` server, which have a `.dwf` suffix). Applications and defaults files do not reference fonts by font file name; they use font names. Each directory for ULTRIX/UWS Version 4.1 fonts has an alias file, called `fonts.alias`. By default, this file uses a special notation (FILE_NAMES_ALIASES) to define the name of each `.pcf` file in the directory as an alias for the font it contains. For example, the font whose name is

```
DEC-ADOBE-Helvetica-Bold-R-Normal--12-120-75-75-P-70-DEC-ISOLATIN1
```

is stored in the file

```
/usr/lib/X11/fonts/decwin/75dpi/helvetica12.pcf
```

The font can be referred to in Xlib routines or defaults files as either

```
DEC-ADOBE-Helvetica-Bold-R-Normal--12-120-75-75-P-70-DEC-ISOLATIN1
```

or

```
helvetica12
```

You can set up additional aliases in the `fonts.alias` file. For more information, refer to the `dxmkfontdir(1X)` reference page and the example file in the next section.

4.3.6.4 Example Font Aliases File – This example file illustrates a workaround for interoperability problems that might exist when Digital applications are used with other non-Digital X servers.

The workaround maps fonts used by Digital's applications to fonts supplied on the MIT X11 Release 4 tape. Therefore, the following font families must already be installed on the server for this workaround to be successful:

- Courier
- Helvetica
- New Century Schoolbook
- Symbol
- Terminal
- Times

This font aliases file example works with any MIT-based X server (just about all X servers).

Note

Use of non-Digital X servers with Digital applications is not supported by Digital.

```
-Adobe-"ITC Avant Garde Gothic"-Book-R-Normal--10-100-75-75-P-59-ISO8859-1
-Adobe-Helvetica-Medium-R-Normal--10-100-75-75-P-56-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-R-Normal--12-120-75-75-P-70-ISO8859-1
-Adobe-Helvetica-Medium-R-Normal--12-120-75-75-P-67-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-R-Normal--14-140-75-75-P-80-ISO8859-1
-Adobe-Helvetica-Medium-R-Normal--14-140-75-75-P-77-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-R-Normal--18-180-75-75-P-103-ISO8859-1
-Adobe-Helvetica-Medium-R-Normal--18-180-75-75-P-98-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-R-Normal--24-240-75-75-P-138-ISO8859-1
-Adobe-Helvetica-Medium-R-Normal--24-240-75-75-P-130-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-R-Normal--8-80-75-75-P-49-ISO8859-1
-Adobe-Helvetica-Medium-R-Normal--8-80-75-75-P-46-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-O-Normal--10-100-75-75-P-59-ISO8859-1
-Adobe-Helvetica-Medium-O-Normal--10-100-75-75-P-57-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-O-Normal--12-120-75-75-P-69-ISO8859-1
-Adobe-Helvetica-Medium-O-Normal--12-120-75-75-P-67-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-O-Normal--14-140-75-75-P-81-ISO8859-1
-Adobe-Helvetica-Medium-O-Normal--14-140-75-75-P-78-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-O-Normal--18-180-75-75-P-103-ISO8859-1
-Adobe-Helvetica-Medium-O-Normal--18-180-75-75-P-98-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-O-Normal--24-240-75-75-P-138-ISO8859-1
-Adobe-Helvetica-Medium-O-Normal--24-240-75-75-P-130-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-O-Normal--8-80-75-75-P-49-ISO8859-1
-Adobe-Helvetica-Medium-O-Normal--8-80-75-75-P-47-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-R-Normal--10-100-75-75-P-61-ISO8859-1
-Adobe-Helvetica-Bold-R-Normal--10-100-75-75-P-60-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-R-Normal--12-120-75-75-P-70-ISO8859-1
-Adobe-Helvetica-Bold-R-Normal--12-120-75-75-P-70-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-R-Normal--14-140-75-75-P-82-ISO8859-1
```

```
-Adobe-Helvetica-Bold-R-Normal--14-140-75-75-P-82-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-R-Normal--18-180-75-75-P-105-ISO8859-1
-Adobe-Helvetica-Bold-R-Normal--18-180-75-75-P-103-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-R-Normal--24-240-75-75-P-140-ISO8859-1
-Adobe-Helvetica-Bold-R-Normal--24-240-75-75-P-138-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-R-Normal--8-80-75-75-P-51-ISO8859-1
-Adobe-Helvetica-Bold-R-Normal--8-80-75-75-P-50-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-O-Normal--10-100-75-75-P-61-ISO8859-1
-Adobe-Helvetica-Bold-O-Normal--10-100-75-75-P-60-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-O-Normal--12-120-75-75-P-71-ISO8859-1
-Adobe-Helvetica-Bold-O-Normal--12-120-75-75-P-69-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-O-Normal--14-140-75-75-P-82-ISO8859-1
-Adobe-Helvetica-Bold-O-Normal--14-140-75-75-P-82-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-O-Normal--18-180-75-75-P-103-ISO8859-1
-Adobe-Helvetica-Bold-O-Normal--18-180-75-75-P-104-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-O-Normal--24-240-75-75-P-139-ISO8859-1
-Adobe-Helvetica-Bold-O-Normal--24-240-75-75-P-138-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-O-Normal--8-80-75-75-P-51-ISO8859-1
-Adobe-Helvetica-Bold-O-Normal--8-80-75-75-P-50-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-R-Normal--10-100-75-75-P-60-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-R-Normal--10-100-75-75-P-60-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-R-Normal--12-120-75-75-P-70-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-R-Normal--12-120-75-75-P-70-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-R-Normal--14-140-75-75-P-81-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-R-Normal--14-140-75-75-P-82-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-R-Normal--18-180-75-75-P-106-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-R-Normal--18-180-75-75-P-103-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-R-Normal--24-240-75-75-P-139-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-R-Normal--24-240-75-75-P-137-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-R-Normal--8-80-75-75-P-50-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-R-Normal--8-80-75-75-P-50-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-O-Normal--10-100-75-75-P-60-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-I-Normal--10-100-75-75-P-60-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-O-Normal--12-120-75-75-P-70-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-I-Normal--12-120-75-75-P-70-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-O-Normal--14-140-75-75-P-82-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-I-Normal--14-140-75-75-P-81-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-O-Normal--18-180-75-75-P-105-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-I-Normal--18-180-75-75-P-104-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-O-Normal--24-240-75-75-P-140-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-I-Normal--24-240-75-75-P-136-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-O-Normal--8-80-75-75-P-50-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-I-Normal--8-80-75-75-P-50-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-R-Normal--10-100-75-75-P-61-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-R-Normal--10-100-75-75-P-66-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-R-Normal--12-120-75-75-P-73-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-R-Normal--12-120-75-75-P-77-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-R-Normal--14-140-75-75-P-85-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-R-Normal--14-140-75-75-P-87-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-R-Normal--18-180-75-75-P-109-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-R-Normal--18-180-75-75-P-113-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-R-Normal--24-240-75-75-P-144-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-R-Normal--24-240-75-75-P-149-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-R-Normal--8-80-75-75-P-51-ISO8859-1
```

```
-Adobe-"New Century Schoolbook"-Bold-R-Normal--8-80-75-75-P-56-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-O-Normal--10-100-75-75-P-62-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-I-Normal--10-100-75-75-P-66-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-O-Normal--12-120-75-75-P-74-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-I-Normal--12-120-75-75-P-76-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-O-Normal--14-140-75-75-P-85-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-I-Normal--14-140-75-75-P-88-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-O-Normal--18-180-75-75-P-109-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-I-Normal--18-180-75-75-P-111-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-O-Normal--24-240-75-75-P-144-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-I-Normal--24-240-75-75-P-148-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-O-Normal--8-80-75-75-P-52-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-I-Normal--8-80-75-75-P-56-ISO8859-1
-"Bigelow & Holmes"-Menu-Medium-R-Normal--10-100-75-75-P-56-ISO8859-1
-Adobe-Helvetica-Bold-R-Normal--10-100-75-75-P-60-ISO8859-1
-"Bigelow & Holmes"-Menu-Medium-R-Normal--12-120-75-75-P-70-ISO8859-1
-Adobe-Helvetica-Bold-R-Normal--12-120-75-75-P-70-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-R-Normal--10-100-75-75-P-62-ISO8859-1
-Adobe-Times-Bold-R-Normal--10-100-75-75-P-57-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-R-Normal--12-120-75-75-P-75-ISO8859-1
-Adobe-Times-Bold-R-Normal--12-120-75-75-P-67-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-R-Normal--14-140-75-75-P-90-ISO8859-1
-Adobe-Times-Bold-R-Normal--14-140-75-75-P-77-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-R-Normal--18-180-75-75-P-112-ISO8859-1
-Adobe-Times-Bold-R-Normal--18-180-75-75-P-99-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-R-Normal--24-240-75-75-P-149-ISO8859-1
-Adobe-Times-Bold-R-Normal--24-240-75-75-P-132-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-R-Normal--8-80-75-75-P-52-ISO8859-1
-Adobe-Times-Bold-R-Normal--8-80-75-75-P-47-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-I-Normal--10-100-75-75-P-67-ISO8859-1
-Adobe-Times-Bold-I-Normal--10-100-75-75-P-57-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-I-Normal--12-120-75-75-P-78-ISO8859-1
-Adobe-Times-Bold-I-Normal--12-120-75-75-P-68-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-I-Normal--14-140-75-75-P-92-ISO8859-1
-Adobe-Times-Bold-I-Normal--14-140-75-75-P-77-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-I-Normal--18-180-75-75-P-115-ISO8859-1
-Adobe-Times-Bold-I-Normal--18-180-75-75-P-98-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-I-Normal--24-240-75-75-P-154-ISO8859-1
-Adobe-Times-Bold-I-Normal--24-240-75-75-P-128-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-I-Normal--8-80-75-75-P-57-ISO8859-1
-Adobe-Times-Bold-I-Normal--8-80-75-75-P-47-ISO8859-1
-Adobe-"ITC Souvenir"-Light-R-Normal--10-100-75-75-P-56-ISO8859-1
-Adobe-Times-Medium-R-Normal--10-100-75-75-P-54-ISO8859-1
-Adobe-"ITC Souvenir"-Light-R-Normal--12-120-75-75-P-68-ISO8859-1
-Adobe-Times-Medium-R-Normal--12-120-75-75-P-64-ISO8859-1
-Adobe-"ITC Souvenir"-Light-R-Normal--14-140-75-75-P-79-ISO8859-1
-Adobe-Times-Medium-R-Normal--14-140-75-75-P-74-ISO8859-1
-Adobe-"ITC Souvenir"-Light-R-Normal--18-180-75-75-P-102-ISO8859-1
-Adobe-Times-Medium-R-Normal--18-180-75-75-P-94-ISO8859-1
-Adobe-"ITC Souvenir"-Light-R-Normal--24-240-75-75-P-135-ISO8859-1
-Adobe-Times-Medium-R-Normal--24-240-75-75-P-124-ISO8859-1
-Adobe-"ITC Souvenir"-Light-R-Normal--8-80-75-75-P-46-ISO8859-1
-Adobe-Times-Medium-R-Normal--8-80-75-75-P-44-ISO8859-1
-Adobe-"ITC Souvenir"-Light-I-Normal--10-100-75-75-P-59-ISO8859-1
```

```
-Adobe-Times-Medium-I-Normal--10-100-75-75-P-52-ISO8859-1
-Adobe-"ITC Souvenir"-Light-I-Normal--12-120-75-75-P-69-ISO8859-1
-Adobe-Times-Medium-I-Normal--12-120-75-75-P-63-ISO8859-1
-Adobe-"ITC Souvenir"-Light-I-Normal--14-140-75-75-P-82-ISO8859-1
-Adobe-Times-Medium-I-Normal--14-140-75-75-P-73-ISO8859-1
-Adobe-"ITC Souvenir"-Light-I-Normal--18-180-75-75-P-104-ISO8859-1
-Adobe-Times-Medium-I-Normal--18-180-75-75-P-94-ISO8859-1
-Adobe-"ITC Souvenir"-Light-I-Normal--24-240-75-75-P-139-ISO8859-1
-Adobe-Times-Medium-I-Normal--24-240-75-75-P-125-ISO8859-1
-Adobe-"ITC Souvenir"-Light-I-Normal--8-80-75-75-P-49-ISO8859-1
-Adobe-Times-Medium-I-Normal--8-80-75-75-P-42-ISO8859-1
-DEC-Terminal-Medium-R-Normal--14-140-75-75-C-8-ISO8859-1
-DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-ISO8859-1
-Bitstream-Terminal-Medium-R-Normal--18-180-75-75-C-11-ISO8859-1
-DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-ISO8859-1
-DEC-Terminal-Medium-R-Normal--28-280-75-75-C-16-ISO8859-1
-DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-ISO8859-1
-Bitstream-Terminal-Medium-R-Normal--36-360-75-75-C-22-ISO8859-1
-DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-ISO8859-1
-DEC-Terminal-Bold-R-Normal--14-140-75-75-C-8-ISO8859-1
-DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-ISO8859-1
-Bitstream-Terminal-Bold-R-Normal--18-180-75-75-C-11-ISO8859-1
-DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-ISO8859-1
-DEC-Terminal-Bold-R-Normal--28-280-75-75-C-16-ISO8859-1
-DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-ISO8859-1
-Bitstream-Terminal-Bold-R-Normal--36-360-75-75-C-22-ISO8859-1
-DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-ISO8859-1
-DEC-Terminal-Bold-R-"Double Wide"--14-140-75-75-C-16-ISO8859-1
-DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-ISO8859-1
-Bitstream-Terminal-Bold-R-"Double Wide"--18-180-75-75-C-22-ISO8859-1
-DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-ISO8859-1
-DEC-Terminal-Bold-R-"Double Wide"--14-140-75-75-C-16
-DEC-DECtech -DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-DEC-DECtech
-Bitstream-Terminal-Bold-R-"Double Wide"--18-180-75-75-C-22
-DEC-DECtech -DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-DEC-DECtech
-DEC-Terminal-Bold-R-Normal--14-140-75-75-C-8
-DEC-DECtech -DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-DEC-DECtech
-Bitstream-Terminal-Bold-R-Normal--18-180-75-75-C-11
-DEC-DECtech -DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-DEC-DECtech
-DEC-Terminal-Bold-R-Normal--28-280-75-75-C-16
-DEC-DECtech -DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-DEC-DECtech
-Bitstream-Terminal-Bold-R-Normal--36-360-75-75-C-22
-DEC-DECtech -DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-DEC-DECtech
-DEC-Terminal-Bold-R-Narrow--14-140-75-75-C-6-ISO8859-1
-DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-ISO8859-1
-Bitstream-Terminal-Bold-R-Narrow--18-180-75-75-C-7-ISO8859-1
-DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-ISO8859-1
-DEC-Terminal-Bold-R-Narrow--28-280-75-75-C-12-ISO8859-1
-DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-ISO8859-1
-Bitstream-Terminal-Bold-R-Narrow--36-360-75-75-C-14-ISO8859-1
-DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-ISO8859-1
-DEC-Terminal-Bold-R-Narrow--14-140-75-75-C-6-DEC-DECtech
-DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-DEC-DECtech
-Bitstream-Terminal-Bold-R-Narrow--18-180-75-75-C-7
```

```
-DEC-DECtech -DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-DEC-DECtech
-DEC-Terminal-Bold-R-Narrow--28-280-75-75-C-12
-DEC-DECtech -DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-DEC-DECtech
-Bitstream-Terminal-Bold-R-Narrow--36-360-75-75-C-14
-DEC-DECtech -DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-DEC-DECtech
-DEC-Terminal-Bold-R-Wide--14-140-75-75-C-12-ISO8859-1
-DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-ISO8859-1
-Bitstream-Terminal-Bold-R-Wide--18-180-75-75-C-14-ISO8859-1
-DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-ISO8859-1
-DEC-Terminal-Bold-R-Wide--14-140-75-75-C-12
-DEC-DECtech -DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-DEC-DECtech
-Bitstream-Terminal-Bold-R-Wide--18-180-75-75-C-14
-DEC-DECtech -DEC-Terminal-Bold-R-Normal--14-140-75-75-C-80-DEC-DECtech
-DEC-Terminal-Medium-R-"Double Wide"--14-140-75-75-C-16-ISO8859-1
-DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-ISO8859-1
-Bitstream-Terminal-Medium-R-"Double Wide"--18-180-75-75-C-22-ISO8859-1
-DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-ISO8859-1
-DEC-Terminal-Medium-R-"Double Wide"--14-140-75-75-C-16
-DEC-DECtech -DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-DEC-DECtech
-Bitstream-Terminal-Medium-R-"Double Wide"--18-180-75-75-C-22
-DEC-DECtech -DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-DEC-DECtech
-DEC-Terminal-Medium-R-Normal--14-140-75-75-C-8
-DEC-DECtech -DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-DEC-DECtech
-Bitstream-Terminal-Medium-R-Normal--18-180-75-75-C-11
-DEC-DECtech -DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-DEC-DECtech
-DEC-Terminal-Medium-R-Normal--28-280-75-75-C-16
-DEC-DECtech -DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-DEC-DECtech
-Bitstream-Terminal-Medium-R-Normal--36-360-75-75-C-22
-DEC-DECtech -DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-DEC-DECtech
-DEC-Terminal-Medium-R-Narrow--14-140-75-75-C-6-ISO8859-1
-DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-ISO8859-1
-Bitstream-Terminal-Medium-R-Narrow--18-180-75-75-C-7-ISO8859-1
-DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-ISO8859-1
-DEC-Terminal-Medium-R-Narrow--28-280-75-75-C-12-ISO8859-1
-DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-ISO8859-1
-Bitstream-Terminal-Medium-R-Narrow--36-360-75-75-C-14-ISO8859-1
-DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-ISO8859-1
-DEC-Terminal-Medium-R-Narrow--14-140-75-75-C-6
-DEC-DECtech -DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-DEC-DECtech
-Bitstream-Terminal-Medium-R-Narrow--18-180-75-75-C-7
-DEC-DECtech -DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-DEC-DECtech
-DEC-Terminal-Medium-R-Narrow--28-280-75-75-C-12
-DEC-DECtech -DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-DEC-DECtech
-Bitstream-Terminal-Medium-R-Narrow--36-360-75-75-C-14
-DEC-DECtech -DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-DEC-DECtech
-DEC-Terminal-Medium-R-Wide--14-140-75-75-C-12-ISO8859-1
-DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-ISO8859-1
-Bitstream-Terminal-Medium-R-Wide--18-180-75-75-C-14-ISO8859-1
-DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-ISO8859-1
-DEC-Terminal-Medium-R-Wide--14-140-75-75-C-12
-DEC-DECtech -DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-DEC-DECtech
-Bitstream-Terminal-Medium-R-Wide--18-180-75-75-C-14
-DEC-DECtech -DEC-Terminal-Medium-R-Normal--14-140-75-75-C-80-DEC-DECtech
-Adobe-"ITC Avant Garde Gothic"-Book-R-Normal--14-100-100-100-P-80-ISO8859-1
```

```
-Adobe-Helvetica-Medium-R-Normal--14-100-100-100-P-76-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-R-Normal--17-120-100-100-P-93-ISO8859-1
-Adobe-Helvetica-Medium-R-Normal--17-120-100-100-P-88-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-R-Normal--20-140-100-100-P-104-ISO8859-1
-Adobe-Helvetica-Medium-R-Normal--20-140-100-100-P-100-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-R-Normal--25-180-100-100-P-138-ISO8859-1
-Adobe-Helvetica-Medium-R-Normal--25-180-100-100-P-130-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-R-Normal--34-240-100-100-P-183-ISO8859-1
-Adobe-Helvetica-Medium-R-Normal--34-240-100-100-P-176-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-R-Normal--11-80-100-100-P-59-ISO8859-1
-Adobe-Helvetica-Medium-R-Normal--11-80-100-100-P-56-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-O-Normal--14-100-100-100-P-81-ISO8859-1
-Adobe-Helvetica-Medium-O-Normal--14-100-100-100-P-78-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-O-Normal--17-120-100-100-P-92-ISO8859-1
-Adobe-Helvetica-Medium-O-Normal--17-120-100-100-P-88-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-O-Normal--20-140-100-100-P-103-ISO8859-1
-Adobe-Helvetica-Medium-O-Normal--20-140-100-100-P-98-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-O-Normal--25-180-100-100-P-138-ISO8859-1
-Adobe-Helvetica-Medium-O-Normal--25-180-100-100-P-130-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-O-Normal--34-240-100-100-P-184-ISO8859-1
-Adobe-Helvetica-Medium-O-Normal--34-240-100-100-P-176-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Book-O-Normal--10-80-100-100-P-59-ISO8859-1
-Adobe-Helvetica-Medium-O-Normal--11-80-100-100-P-57-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-R-Normal--14-100-100-100-P-82-ISO8859-1
-Adobe-Helvetica-Bold-R-Normal--14-100-100-100-P-82-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-R-Normal--17-120-100-100-P-93-ISO8859-1
-Adobe-Helvetica-Bold-R-Normal--17-120-100-100-P-92-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-R-Normal--20-140-100-100-P-105-ISO8859-1
-Adobe-Helvetica-Bold-R-Normal--20-140-100-100-P-105-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-R-Normal--25-180-100-100-P-140-ISO8859-1
-Adobe-Helvetica-Bold-R-Normal--25-180-100-100-P-138-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-R-Normal--34-240-100-100-P-182-ISO8859-1
-Adobe-Helvetica-Bold-R-Normal--34-240-100-100-P-182-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-R-Normal--11-80-100-100-P-61-ISO8859-1
-Adobe-Helvetica-Bold-R-Normal--11-80-100-100-P-60-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-O-Normal--14-100-100-100-P-82-ISO8859-1
-Adobe-Helvetica-Bold-O-Normal--14-100-100-100-P-82-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-O-Normal--17-120-100-100-P-93-ISO8859-1
-Adobe-Helvetica-Bold-O-Normal--17-120-100-100-P-92-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-O-Normal--20-140-100-100-P-103-ISO8859-1
-Adobe-Helvetica-Bold-O-Normal--20-140-100-100-P-103-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-O-Normal--25-180-100-100-P-139-ISO8859-1
-Adobe-Helvetica-Bold-O-Normal--25-180-100-100-P-138-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-O-Normal--34-240-100-100-P-183-ISO8859-1
-Adobe-Helvetica-Bold-O-Normal--34-240-100-100-P-182-ISO8859-1
-Adobe-"ITC Avant Garde Gothic"-Demi-O-Normal--11-80-100-100-P-61-ISO8859-1
-Adobe-Helvetica-Bold-O-Normal--11-80-100-100-P-60-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-R-Normal--14-100-100-100-P-81-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-R-Normal--14-100-100-100-P-82-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-R-Normal--17-120-100-100-P-89-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-R-Normal--17-120-100-100-P-91-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-R-Normal--19-140-100-100-P-106-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-R-Normal--20-140-100-100-P-103-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-R-Normal--24-180-100-100-P-139-ISO8859-1
```

```
-Adobe-"New Century Schoolbook"-Medium-R-Normal--25-180-100-100-P-136-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-R-Normal--33-240-100-100-P-180-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-R-Normal--34-240-100-100-P-181-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-R-Normal--11-80-100-100-P-60-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-R-Normal--11-80-100-100-P-60-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-O-Normal--14-100-100-100-P-82-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-I-Normal--14-100-100-100-P-81-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-O-Normal--19-120-100-100-P-89-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-I-Normal--17-120-100-100-P-92-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-O-Normal--20-140-100-100-P-105-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-I-Normal--20-140-100-100-P-104-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-O-Normal--24-180-100-100-P-140-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-I-Normal--25-180-100-100-P-136-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-O-Normal--33-240-100-100-P-181-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-I-Normal--34-240-100-100-P-182-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Book-O-Normal--11-80-100-100-P-60-ISO8859-1
-Adobe-"New Century Schoolbook"-Medium-I-Normal--11-80-100-100-P-60-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-R-Normal--14-100-100-100-P-85-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-R-Normal--14-100-100-100-P-87-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-R-Normal--17-120-100-100-P-92-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-R-Normal--17-120-100-100-P-99-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-R-Normal--19-140-100-100-P-109-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-R-Normal--20-140-100-100-P-113-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-R-Normal--24-180-100-100-P-144-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-R-Normal--25-180-100-100-P-149-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-R-Normal--33-240-100-100-P-184-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-R-Normal--34-240-100-100-P-193-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-R-Normal--11-80-100-100-P-61-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-R-Normal--11-80-100-100-P-66-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-O-Normal--14-100-100-100-P-85-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-I-Normal--14-100-100-100-P-88-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-O-Normal--17-120-100-100-P-92-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-I-Normal--17-120-100-100-P-99-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-O-Normal--19-140-100-100-P-109-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-I-Normal--20-140-100-100-P-111-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-O-Normal--24-180-100-100-P-144-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-I-Normal--25-180-100-100-P-148-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-O-Normal--33-240-100-100-P-184-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-I-Normal--34-240-100-100-P-193-ISO8859-1
-Adobe-"ITC Lubalin Graph"-Demi-O-Normal--11-80-100-100-P-62-ISO8859-1
-Adobe-"New Century Schoolbook"-Bold-I-Normal--11-80-100-100-P-66-ISO8859-1
-"Bigelow & Holmes"-Menu-Medium-R-Normal--13-100-100-100-P-77-ISO8859-1
-Adobe-Helvetica-Bold-R-Normal--14-100-100-100-P-82-ISO8859-1
-"Bigelow & Holmes"-Menu-Medium-R-Normal--16-120-100-100-P-92-ISO8859-1
-Adobe-Helvetica-Bold-R-Normal--17-120-100-100-P-92-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-R-Normal--14-100-100-100-P-90-ISO8859-1
-Adobe-Times-Bold-R-Normal--14-100-100-100-P-76-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-R-Normal--17-120-100-100-P-94-ISO8859-1
-Adobe-Times-Bold-R-Normal--17-120-100-100-P-88-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-R-Normal--20-140-100-100-P-112-ISO8859-1
-Adobe-Times-Bold-R-Normal--20-140-100-100-P-100-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-R-Normal--25-180-100-100-P-149-ISO8859-1
-Adobe-Times-Bold-R-Normal--25-180-100-100-P-132-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-R-Normal--34-240-100-100-P-191-ISO8859-1
```

```
-Adobe-Times-Bold-R-Normal--34-240-100-100-P-177-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-R-Normal--11-80-100-100-P-62-ISO8859-1
-Adobe-Times-Bold-R-Normal--11-80-100-100-P-57-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-I-Normal--14-100-100-100-P-92-ISO8859-1
-Adobe-Times-Bold-I-Normal--14-100-100-100-P-77-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-I-Normal--17-120-100-100-P-98-ISO8859-1
-Adobe-Times-Bold-I-Normal--17-120-100-100-P-86-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-I-Normal--20-140-100-100-P-115-ISO8859-1
-Adobe-Times-Bold-I-Normal--20-140-100-100-P-98-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-I-Normal--25-180-100-100-P-154-ISO8859-1
-Adobe-Times-Bold-I-Normal--25-180-100-100-P-128-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-I-Normal--34-240-100-100-P-197-ISO8859-1
-Adobe-Times-Bold-I-Normal--34-240-100-100-P-170-ISO8859-1
-Adobe-"ITC Souvenir"-Demi-I-Normal--11-80-100-100-P-67-ISO8859-1
-Adobe-Times-Bold-I-Normal--11-80-100-100-P-57-ISO8859-1
-Adobe-"ITC Souvenir"-Light-R-Normal--14-100-100-100-P-79-ISO8859-1
-Adobe-Times-Medium-R-Normal--14-100-100-100-P-74-ISO8859-1
-Adobe-"ITC Souvenir"-Light-R-Normal--17-120-100-100-P-85-ISO8859-1
-Adobe-Times-Medium-R-Normal--17-120-100-100-P-84-ISO8859-1
-Adobe-"ITC Souvenir"-Light-R-Normal--20-140-100-100-P-102-ISO8859-1
-Adobe-Times-Medium-R-Normal--20-140-100-100-P-96-ISO8859-1
-Adobe-"ITC Souvenir"-Light-R-Normal--25-180-100-100-P-135-ISO8859-1
-Adobe-Times-Medium-R-Normal--25-180-100-100-P-125-ISO8859-1
-Adobe-"ITC Souvenir"-Light-R-Normal--34-240-100-100-P-174-ISO8859-1
-Adobe-Times-Medium-R-Normal--34-240-100-100-P-170-ISO8859-1
-Adobe-"ITC Souvenir"-Light-R-Normal--11-80-100-100-P-56-ISO8859-1
-Adobe-Times-Medium-R-Normal--11-80-100-100-P-54-ISO8859-1
-Adobe-"ITC Souvenir"-Light-I-Normal--14-100-100-100-P-82-ISO8859-1
-Adobe-Times-Medium-I-Normal--14-100-100-100-P-73-ISO8859-1
-Adobe-"ITC Souvenir"-Light-I-Normal--17-120-100-100-P-88-ISO8859-1
-Adobe-Times-Medium-I-Normal--17-120-100-100-P-84-ISO8859-1
-Adobe-"ITC Souvenir"-Light-I-Normal--20-140-100-100-P-104-ISO8859-1
-Adobe-Times-Medium-I-Normal--20-140-100-100-P-94-ISO8859-1
-Adobe-"ITC Souvenir"-Light-I-Normal--25-180-100-100-P-139-ISO8859-1
-Adobe-Times-Medium-I-Normal--25-180-100-100-P-125-ISO8859-1
-Adobe-"ITC Souvenir"-Light-I-Normal--34-240-100-100-P-177-ISO8859-1
-Adobe-Times-Medium-I-Normal--34-240-100-100-P-168-ISO8859-1
-Adobe-"ITC Souvenir"-Light-I-Normal--11-80-100-100-P-59-ISO8859-1
-Adobe-Times-Medium-I-Normal--11-80-100-100-P-52-ISO8859-1
-DEC-Terminal-Medium-R-Normal--14-100-100-100-C-8-ISO8859-1
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-ISO8859-1
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-11-ISO8859-1
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-ISO8859-1
-DEC-Terminal-Medium-R-Normal--28-200-100-100-C-16-ISO8859-1
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-ISO8859-1
-Bitstream-Terminal-Medium-R-Normal--36-280-100-100-C-22-ISO8859-1
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-ISO8859-1
-DEC-Terminal-Bold-R-Normal--14-100-100-100-C-8-ISO8859-1
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-ISO8859-1
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-11-ISO8859-1
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-ISO8859-1
-DEC-Terminal-Bold-R-Normal--28-200-100-100-C-16-ISO8859-1
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-ISO8859-1
-Bitstream-Terminal-Bold-R-Normal--36-280-100-100-C-22-ISO8859-1
```

```
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-ISO8859-1
-DEC-Terminal-Bold-R-"Double Wide"--14-100-100-100-C-16-ISO8859-1
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-ISO8859-1
-Bitstream-Terminal-Bold-R-"Double Wide"--18-140-100-100-C-22-ISO8859-1
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-ISO8859-1
-DEC-Terminal-Bold-R-"Double Wide"--14-100-100-100-C-16-DEC-DECtech
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-DEC-DECtech
-Bitstream-Terminal-Bold-R-"Double Wide"--18-140-100-100-C-22-DEC-DECtech
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-DEC-DECtech
-DEC-Terminal-Bold-R-Normal--14-100-100-100-C-8-DEC-DECtech
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-DEC-DECtech
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-11-DEC-DECtech
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-DEC-DECtech
-DEC-Terminal-Bold-R-Normal--28-200-100-100-C-16-DEC-DECtech
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-DEC-DECtech
-Bitstream-Terminal-Bold-R-Normal--36-280-100-100-C-22-DEC-DECtech
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-DEC-DECtech
-DEC-Terminal-Bold-R-Narrow--14-100-100-100-C-6-ISO8859-1
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-ISO8859-1
-Bitstream-Terminal-Bold-R-Narrow--18-140-100-100-C-7-ISO8859-1
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-ISO8859-1
-DEC-Terminal-Bold-R-Narrow--28-200-100-100-C-12-ISO8859-1
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-ISO8859-1
-Bitstream-Terminal-Bold-R-Narrow--36-280-100-100-C-14-ISO8859-1
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-ISO8859-1
-DEC-Terminal-Bold-R-Narrow--14-100-100-100-C-6-DEC-DECtech
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-DEC-DECtech
-Bitstream-Terminal-Bold-R-Narrow--18-140-100-100-C-7-DEC-DECtech
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-DEC-DECtech
-DEC-Terminal-Bold-R-Narrow--28-200-100-100-C-12-DEC-DECtech
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-DEC-DECtech
-Bitstream-Terminal-Bold-R-Narrow--36-280-100-100-C-14-DEC-DECtech
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-DEC-DECtech
-DEC-Terminal-Bold-R-Wide--14-100-100-100-C-12-ISO8859-1
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-ISO8859-1
-Bitstream-Terminal-Bold-R-Wide--18-140-100-100-C-14-ISO8859-1
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-ISO8859-1
-DEC-Terminal-Bold-R-Wide--14-100-100-100-C-12-DEC-DECtech
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-DEC-DECtech
-Bitstream-Terminal-Bold-R-Wide--18-140-100-100-C-14-DEC-DECtech
-Bitstream-Terminal-Bold-R-Normal--18-140-100-100-C-110-DEC-DECtech
-DEC-Terminal-Medium-R-"Double Wide"--14-100-100-100-C-16-ISO8859-1
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-ISO8859-1
-Bitstream-Terminal-Medium-R-"Double Wide"--18-140-100-100-C-22-ISO8859-1
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-ISO8859-1
-DEC-Terminal-Medium-R-"Double Wide"--14-100-100-100-C-16-DEC-DECtech
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-DEC-DECtech
-Bitstream-Terminal-Medium-R-"Double Wide"--18-140-100-100-C-22-DEC-DECtech
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-DEC-DECtech
-DEC-Terminal-Medium-R-Normal--14-100-100-100-C-8-DEC-DECtech
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-DEC-DECtech
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-11-DEC-DECtech
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-DEC-DECtech
-DEC-Terminal-Medium-R-Normal--28-200-100-100-C-16-DEC-DECtech
```

```
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-DEC-DECtech
-Bitstream-Terminal-Medium-R-Normal--36-280-100-100-C-22-DEC-DECtech
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-DEC-DECtech
-DEC-Terminal-Medium-R-Narrow--14-100-100-100-C-6-ISO8859-1
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-ISO8859-1
-Bitstream-Terminal-Medium-R-Narrow--18-140-100-100-C-7-ISO8859-1
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-ISO8859-1
-DEC-Terminal-Medium-R-Narrow--28-200-100-100-C-12-ISO8859-1
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-ISO8859-1
-Bitstream-Terminal-Medium-R-Narrow--36-280-100-100-C-14-ISO8859-1
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-ISO8859-1
-DEC-Terminal-Medium-R-Narrow--14-100-100-100-C-6-DEC-DECtech
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-DEC-DECtech
-Bitstream-Terminal-Medium-R-Narrow--18-140-100-100-C-7-DEC-DECtech
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-DEC-DECtech
-DEC-Terminal-Medium-R-Narrow--28-200-100-100-C-12-DEC-DECtech
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-DEC-DECtech
-Bitstream-Terminal-Medium-R-Narrow--36-280-100-100-C-14-DEC-DECtech
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-DEC-DECtech
-DEC-Terminal-Medium-R-Wide--14-100-100-100-C-12-ISO8859-1
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-ISO8859-1
-Bitstream-Terminal-Medium-R-Wide--18-140-100-100-C-14-ISO8859-1
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-ISO8859-1
-DEC-Terminal-Medium-R-Wide--14-100-100-100-C-12-DEC-DECtech
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-DEC-DECtech
-Bitstream-Terminal-Medium-R-Wide--18-140-100-100-C-14-DEC-DECtech
-Bitstream-Terminal-Medium-R-Normal--18-140-100-100-C-110-DEC-DECtech
```

4.3.6.5 Font Properties – The font properties generally include the font name fields and other useful global font information, such as the height of capitals (CAP_HEIGHT), calculated weight and setwidth (WEIGHT and SETWIDTH), and so on. All ULTRIX/UWS Version 4.1 fonts (except Terminal) have at least the following properties:

```
FONT_ASCENT         FONT_DESCENT
DEFAULT_CHAR        X_HEIGHT
WEIGHT              POINT_SIZE
FACE_NAME           COPYRIGHT
FAMILY_NAME         FONT_NAME_REGISTRY
FOUNDRY             WEIGHT_NAME
SLANT               SETWIDTH_NAME
ADD_STYLE_NAME      PIXEL_SIZE
RESOLUTION_X        RESOLUTION_Y
AVERAGE_WIDTH       CHARSET_REGISTRY
CHARSET_ENCODING    CHARSET_COLLECTIONS
CAP_HEIGHT          NOTICE
```

Fonts derived from PostScript printer fonts also have the property _DEC_DEVICE_NAMES. This property sets up the correspondence between printer fonts requested by an X Display PostScript (XDPS) client (such as a Courier font

requested in a PostScript file) and the screen fonts available to the server. During a font lookup, XDPS chooses a font whose property is "PS=Courier" when a client file requests Courier font. Some examples are as follows:

```
_DEC_DEVICE_NAMES "PS=AvantGarde-Demi"
_DEC_DEVICE_NAMES "PS=Century-Schoolbook-Bold-Italic"
```

Any application (and not just Display PostScript applications) can use this property to ensure that a screen font corresponds to a specified PostScript printer font. To do so, the application searches for a screen font with the `_DEC_DEVICE_NAMES` property specifying that printer font.

4.3.6.6 Changes to the Terminal Font – Since ULTRIX/UWS Version 2.1, line-drawing characters (glyphs) no longer reside within the terminal font. They can be found within the `terminal_dectech` font.

The following fonts are for glyphs:

```
/usr/lib/X11/fonts/decwin/75dpi/terminal_dectech18.pcf

/usr/lib/X11/fonts/decwin/75dpi/terminal_bold_dectech18.pcf
```

4.3.6.7 Viewing/Mailing DDIF Files with Missing External References – `DDIF` files (created by applications such as `DECwrite`), may contain references to external files containing fonts or other data. If you try to view a `DDIF` file with any application linked with `libdvr.a` (such as `dxvdoc` and `dxmail`) containing references to external files that cannot be located on your system, the view fails.

The `dxvdoc` application returns a message such as "Unexpected error converting aggregate" or "CDA could not open file". The `dxmail` application displays a blank viewer window, and no error message. If you have an application using `libdvr.a`, the application will receive an error status such as `CDA$_OPENFAIL`.

If you mail a `DECwrite` file that contains references to a system-library style file to a user on a system without DECwrite installed, the mail message will not be viewed successfully on the receiving end. (System-library style files are installed as part of a `DECwrite` installation in `/usr/lib/cda/*.doc_style`.) The user will see a blank viewer window, and no message in `dxmail`.

System-library style files are not packed up in the mail message because they are assumed to be a system resource that is present on each system. Normally, files referenced externally are packed up in the mail, unless the references are stored as "no-copy" such as `DECwrite` references to system-library style files.

To get around the problem, make sure your files with external references have all of their externally referenced files present on the machine on which you wish to view them.

Use the `cdoc` utility to determine if your file refers to external files. The following command converts a `DDIF` file to analysis format, and then searches the analysis output `myfile.doc` for any external reference labels:

```
# cdoc -s ddif -d analysis myfile.doc | grep ERF_LABEL
```

If you find that there are references to files not present on your system, copy over the files to be able to successfully view the entire document. If you are missing `DECwrite` system-library style files, copy them from a system with `DECwrite` installed from `/usr/lib/cda/*.*.doc_style`.

4.4 User Environment

This section contains information on applications connected to the user environment.

4.4.1 Window Manager - dxwm

The following notes apply to the ULTRIX/UWS Version 4.1 DECwindows Window Manager.

4.4.1.1 Delay in Appearance of Windows – The window manager may prevent applications from mapping drawables to the display until an existing application window is unfocused and refocused.

4.4.1.2 Naming Windows and Icons – The `dxwm` window manager currently lets you name windows and icons. Client programs can specify their names by using the XA_WM_NAME and XA_WM_ICON_NAME properties defined by DECwindows.

The `dxwm` window manager uses the values of these properties when it decorates the client window or icon. If the client does not specify a value for the XA_WM_ICON_NAME property, `dxwm` uses the one set for the XA_WM_NAME property for the icon as well.

The following example shows how to define the name of a window by using the Xlib `XChangeProperty` function:

```
main ( ) {
Window win;
int winW, winH;
int winX, winY;
XSetWindowAttributes xswa;
       /*  open the display  */
winW = 600;
winH = 600;
winX = (DisplayWidth(dpy,0)-winW)>>1;
winY = (DisplayHeight(dpy,0)-winW)>>1;
xswa.event_mask = 0
xswa.background_pixel = BlackPixel(dpy,0);
win = XCreateWindow(dpy,RootWindow(dpy,0),winX,winY,winW,winH,0,
       DefaultDepth(dpy,0),InputOutput,DefaultVisual(dpy,0),
       CWEventMask | CWBackPixel, &xswa);
XChangeProperty (dpy,win,XA_WM_NAME,XA_STRING,8,PropModeReplace,
       "My Window",9);
       .
       .
       .
}
```

In this example:

- The XA_WM_NAME property specifies that property is to be changed, while XA_STRING specifies its data type.
- The argument 8 indicates that the data is in 8-bit format.

- The `PropModeReplace` argument indicates that the previous associated information is to be discarded. For further information, see the *Guide to the Xlib Library*.
- The string ''My Window'' is the new name for the XA_WM_NAME property, and the argument 9 indicates that the string has nine characters.

To change the icon name, substitute XA_WM_ICON_NAME for XA_WM_NAME.

4.4.2 Operator Cannot Log in to Session Manager

The operator cannot log in to the operator account from the `Start Session` login prompt. To invoke operator services, the operator can log in to any account, then use the `su` command to log in as `operator.` You can create a special account for the operator or use the root account. Remember to use the `passwd`(1) command to create a password for the operator account. This is necessary even though the operator does not log directly into the operator account.

4.4.3 Delay in an Application's Appearance

Pseudo ttys are used by many types of applications, including terminal emulators and text editors. An application that uses pseudo ttys can take up to five minutes to appear on the display. If an application window does not appear on the display within a minute, start a second version of the same application. The second version of the application usually displays quickly. When the sluggish version of the application finally appears, you can delete it without harming the second version of the application.

Sometimes the first `dxterm` started by the session manager is affected by this problem.

4.4.4 Calendar - dxcalendar

The following notes apply to the `dxcalendar` program:

- There has been a change in the database format of the `dxcalendar` program for VAX systems. The `dxcalendar` program informs you of the format change by asking if you want to save the old file. The new format is equivalent to the DECstation/DECsystem 3100 series processor format.
- Repeat entries are lost when you convert the data file of `dxcalendar` from ULTRIX/UWS Version 2.2 (VAX) to ULTRIX/UWS Version 4.1 (VAX or RISC).
- The `dxcalendar` program allows you to define your working hours (using the Customize dialog box). Defining your working hours has two benefits:
  - When `dxcalendar` starts, it aligns the day display such that the start of your working hours is at the top of the display.
  - A dark line (time bar) is displayed on the left and right side of the day display. On the left side, the time bar is expanded to map your timeslots one-to-one. On the right hand side, the scrollbar represents the 24 hours and the time bar represents your working day.
- `dxcalendar` displays only a few hours at a time in the DayView... dialog box.

To display ten or more hours, use the Day View... dialog box under the Customize menu and change the Increment value to Half Hour. Press the Apply and OK options to confirm your changes. You may wish to resize the Day View to use most or all of the height of the screen. Then select the Save Settings option under the Customize menu to enable the `dxcalendar` application to save the selected settings.

4.4.5 Visual Differences Program - dxdiff

Selecting files in the `dxdiff` program does not automatically cause the differences to be shown. The correct procedure for using the `dxdiff` program is as follows: program to list the differences between two files is as follows:

1. Select file number one (left).
2. Select file number two (right).
3. Select the item Do Differences from the Differences pulldown menu. Note that this may take some time and no clock face icon (an indication that the computer is busy) becomes visible.

4.4.6 DECwindows Debugger - dxdb

The `dxdb` program does not allow you to view a module that has a left bracket character (`[`) as a part of its name.

4.4.7 Mail - dxmail

The `dxmail` application does not notify users of problems viewing compound documents. Using the CDA viewer for example, unresolved external reference problems result in an empty view window being displayed in `dxmail`'s read window. No message is given to the user when a bad file is sent.

In addition, if the `Quit` option is selected from the Continue/Quit/CoreDump&Quit dialog box, all workstation windows will freeze until all `dxmail` windows disappear.

4.4.8 Paint - dxpaint

The following notes pertain to the `dxpaint` program:

- The Select_All (from the Options menu) and Flood Fill (using the Paint Bucket tool) functions operate only on the visible portion of the screen.
- The Options menu has changed slightly in appearance. The toggle for writing mode (either transparent or opaque) has been changed to two radio buttons.
- Support has been included for AIL (Applications Interface Library) allowing `dxpaint` to be called by other applications.
- Support for 300 dpi pictures (pictures larger than the screen) has been included. Previously, `dxpaint` could only edit pictures of size smaller than or equal to the size of the screen. Now pictures of arbitrary size can be edited.
- You can now set output resolution as well as picture size.

- You can now use scroll bars to move to any portion of the picture.
- The Fullview option allows you to display the entire picture scaled down, choose (window sized) the area to move to, and crop the picture to an arbitrary size.
- Keyboard accelerators include:
  - Alt/C = Copy
  - Alt/P = Paint
  - Alt/Q = Quit
  - Alt/S = Save
  - Alt/V = Paste
  - Alt/W = Refresh
  - Alt/X = Cut
  - Alt/Z = Undo
- Select All and Flood Fill act only on the visible portion of the picture and not on the entire picture.
- Transparent/Opaque writing mode affects spraycan, selection and text in addition to what it already affected.
- Scale Picture options allows you to scale the entire picture.
- Brush and Spraycan use the outline pattern instead of the Fill pattern.

4.4.8.1 Drawing Rectangles or Squares Using a Small Line Width – Drawing rectangles (or squares) whose line length is smaller than the specified line width may leave some garbage in the pixmap. If while rubberbanding a rectangle you decide that you do not want it, complete the rectangle by releasing MB1, and then perform an UNDO.

However, if you attempt to rubberband a rectangle whose line is zero width or length, garbage might appear in the pixmap that an UNDO will not clear.

4.4.8.2 Specifying a Tilde (~) as Part of a File Specification – The tilde (~) will not be translated as the user's login directory if it is used as part of the file specification. As a result, if a tilde is used, `dxpaint` will respond with a message that the file could not be saved (or opened). Always use the full path name when opening or saving files.

A tilde is parsed as the user's login directory if it is used as part of the file filter.

4.4.9 PostScript Previewer - dxpsview

The notes in this section pertain the to the PostScript Previewer, `dxpsview`:

4.4.9.1 Scale Factors Larger than 2.0 – Due to a swap space memory limitation, selecting a scale factor greater than 2.0 can crash the X server.

4.4.9.2 PostScript File Identification – Many PostScript files created by document formatters, such as those used with `ditroff`, adhere to the Adobe page description commenting conventions. You can verify that a file is properly commented (and thus positively identifiable as a PostScript file) by checking to see that the first line is

```
%!PS-Adobe-X.0
```

where X.0 is a PostScript version number.

4.4.9.3 Viewing Uncommented PostScript Files – The PostScript Previewer now lets you view files whose first two characters are not

```
%!
```

If you open such a file for viewing, a warning message appears, and you are asked to confirm that the file is a PostScript file.

4.4.10 Session Manager - dxsession

The following notes pertain to the Session Manager, `dxsession`.

4.4.10.1 Pause Feature Does not Use Updated Password – The Pause feature under the session manager's Session menu requires that you enter your login password to regain access to your workstation. If you have changed your password using the `passwd`(1) command since you last invoked the session manager, you must use your old password, rather than your new one, to regain access.

4.4.10.2 Intensity Labels – For some systems, the Red, Green, and Blue intensity labels that appear in the color selection box that you invoke from the Customize Window dialog box are clipped so that only the tops of the letters are visible. To correct the problem, edit the `/usr/lib/X11/app-defaults/SessionManager` file and delete the following three lines:

```
*Color Attributes.RedScale.height
*Color Attributes.GreenScale.height
*Color Attributes.BlueScale.height
```

4.4.10.3 Setting the Window Screen Background Using the Customize Menu – The Window Screen background pattern option under the Customize menu does not work properly. It does not reflect the pattern until the Apply or OK option is selected. It does not update if the Default option is selected.

4.4.11 DECterm Terminal Emulator - dxterm

The following notes pertain to the DECterm terminal emulator, `dxterm`.

4.4.11.1 User-Defined Key Definitions (UDKs) – UDK definitions supported by `dxterm` allow you to define definitions for shifted function keys (F6 to F20, including Help and Do), using escape sequences. This is described in the *DECterm Text Programming Manual* (part of the EK-DECTERM-DK kit, which also includes the *DECterm Graphics Programming Manual*; this kit can be ordered from

DECdirect at 1-800-DIGITAL). UDKs are also described in manuals for video terminals, such as the *VT330/VT340 Programmer Reference Manual*.

Since UDKs use shifted function keys, you can define these keys for any desired purpose without conflicting with the unshifted function keys that are reserved for operating system and application use. You can choose to lock your UDKs through `dxterm` by selecting the General item from the Customize menu, or by using the UDK Definition Device Control String (DCS). Thus, applications cannot count on being able to redefine these keys.

Do not confuse UDKs with operating system features such as the DEFINE/KEY command on the VMS system and `termcaps` on the ULTRIX system. Those features work with unshifted function keys (or in the case of `termcaps`, with any keyboard key that sends a known input sequence). UDKs use the shifted function keys that send input sequences that look to the application and operating system as if they were typed by you.

UDKs work only when `dxterm` is operating in VT300 mode, not VT100 mode or VT52 mode. To check this, go into the Customize General dialog box, select General from the Customize menu, and make sure that the Terminal Mode is VT300 Mode, 7 Bit Controls or VT300 Mode, 8 Bit Controls.

In the description that follows, 8-bit characters are given in terms of their hexadecimal ASCII values using the C notation. For example, `0x41`, decimal 65, is the ASCII code for the letter A. (The 8-bit ASCII character set is given in the *UWS DECwindows Desktop Applications Guide*,"UPSS DEC Supplemental".)

UDK definitions are not typed directly at the shell prompt. (If they were, they would be intercepted by the operating system and not seen by `dxterm`.) They must be output from the computer to `dxterm`. This can be accomplished in a number of ways. One way is to create a file, using any text editor that allows you to enter nonprinting characters such as ESCAPE. Then copy the file to the terminal, using `cat`(1). Another way is to output text strings directly containing the UDK definitions; that is, from a shell script using `echo`(1). An additional way is to output the strings to the terminal using a program. See the examples below.

To define one or more UDKs, use the following escape sequence:

```
DCS        Pc;Pl        |        Ky1/St1;...Kyn/Stn        ST
```

In this sequence:

- `DCS` is the Device Control String Introducer (ASCII `0x90`). In a 7-bit environment, `DCS` can be sent as ESC P (ASCII `0x1b` and ASCII `0x50`, with no space in between)
- `Pc` is the clear parameter. A value of 0 clears all keys before loading new values (that is, sets them all to empty). A value of 1 clears just the keys that you are loading. If you do not specify `Pc`, it defaults to 0 (that is, all keys that are not defined in the device control string are cleared).
- `Pl` is the lock parameter. A value of 0 locks the keys. If you want to load new values into the keys later, you must unlock the keys from the Customize General menu within DECterm. A value of 1 does not lock the keys. The keys are unlocked and can be redefined with another DECUDK string. If you do not specify `P1`, it defaults to 0 (that is, the keys are locked after the device control string takes place).

Note

If Pl is 1 and the keys are already locked, nothing happens. This is because once the keys are locked they can only be unlocked through the Customize General menu in dxterm (that is, they cannot be unlocked from a program).

Note that Pc and P1 are separated by a semicolon.

This sequence clears all UDKs without locking them:

```
DCS 0 ; 1 | ST
```

This sequence locks all UDKs without clearing them:

```
DCS 1 ; 0 | ST
```

- | is the final character (ASCII 0x7c). The final character separates the clear and lock parameters from the key definition string.
- Ky1/St1;...Kyn/Stn are the key definition strings; you can have zero or more of these, each of which defines a single shifted function key. Each string consists of a string selector number (Kyn) and a string parameter (Stn), separated by a slash (ASCII 0x2f). A semicolon (ASCII 0x3b) separates different strings.

Note

You cannot define the shifted function key F11 through dxterm.

The key selector value (Kyn) indicates which key you are defining:

Key	Kyn Value	Key	Kyn Value
F6	17	Help	28
F7	18	Do	29
F8	19		
F9	20	F17	31
F10	21	F18	32
		F19	33
F11	23	F20	34
F12	24		
F13	25		
F14	26		

Note that these are not ASCII codes but digits, so the code for F18, 32, means the digit 3 (ASCII 0x33) followed by the digit 2 (ASCII 0x32).

The string parameter (Stn) for each string definition is the encoded definition of the key being defined; that is, the sequence of ASCII codes that will be sent to the application. String parameters consist of a series of hexadecimal pairs, one pair for each character in the definition. Each hex pair defines an 8-bit character according to its value in the ASCII table; the hex pair can be uppercase (for example, 4E for the letter "N") or lowercase (for example, 4e for the letter "N").

- ST is the string terminator (ASCII 0x9c).

You should consider the following guidelines when loading UDKs:

- Before loading new definitions, it is a good idea to clear the old key definitions without locking them and then load the new definitions in another DECUDK string. This will prevent the memory used for UDK definitions from becoming fragmented.
- If you redefine a key, the old definition is lost. This may free up some space if the new definition uses fewer bytes than the old one.
- There are two ways to lock UDKs, but only one way to unlock them. To lock UDKs you can use the Lock UDKs toggle button in Customize General or a DECUDK control string. To unlock UDKs, you must use the Lock UDKs toggle button.
- The default value for each key definition is empty. When you clear UDKs, they are empty.
- You cannot save UDK definitions using `dxterm`; the definitions are lost when you exit the `dxterm` window. Because of this, it is a good idea to load the key definitions that you want in your login file,
- An invalid hex pair in a DECUDK string stops a UDK load sequence. When a load sequence stops (due to an error or other cause), `dxterm` saves any keys already loaded and displays the rest of the DECUDK sequence on the screen.

The following example is an ULTRIX shell script that demonstrates how to define more than one shifted function key in the same DECUDK Device Control String (DCS). Note that DCSs can continue over more than one line, as shown in this program. This program was written to execute in VT300 mode, 7-bit controls.

This example defines the shifted function key F6 to be `ls -l`<CR>, where <CR> indicates a carriage return. It also defines the shifted function key F7 to be `date`<CR>. In the example that follows ^[is the escape character as it appears when inserted using the text editor `vi`(1). To enter the escape character in `vi`, while in insert mode, first enter CRTL/V then press the Escape key.

After you execute the shell script the shifted functions keys F6 and F7 are defined.

The ULTRIX shell script follows:

```
echo '^[P'                # DCS Introducer
echo '1;'                 # Pc = 1, Clear only keys that are defined
echo '1'                  # Pl = 1, Do not lock the shifted function keys
echo '|'                  # | = Final Character
echo '17/6c73202d6c0d'    # Ky1/St1 = F6/ls -l<CR>
echo ';18/646174650d'     # Ky2/St2 = F7/date<CR>
echo '^[\'                # ST = String Terminator
```

The following example shows how to define these same two keys, function key F6 and function key F7, from a C program instead of a shell script on ULTRIX.

```
#define ESCAPE '\033'

main()
    {

    /*
     * Send the UDK introducer that does not clear or lock UDK
     * definitions.  Remember that the default for the clear and
     * lock parameters is 0, so if these parameters are omitted the
     * UDKs will be both cleared and locked.
     */
```

```
    printf( "%cP1;1|", ESCAPE );

    /*
     * Define shift-F6 to be "ls -l" terminated with a Return.
     */

    printf( "17/6c73202d6c0d" );

    /*
     * Define shift-F7 to be "date" terminated with a Return.
     */

    printf( ";18/646174650d" );

   /*
    * Terminate the DECUDK command with <ESC>\ and send a newline.
    */

    printf( "%c\\\n", ESCAPE );

    }
```

In this example the final `printf` was terminated with a line feed, but in fact line feeds and spaces could have been inserted at any point in the DECUDK device control string except for within the introducer sequence (in this case <ESC>P|). Note that 8-bit characters can be defined as well as 7-bit characters, although 7-bit ASCII has been used for convenience in the above examples.

4.4.11.2 Command-Line Resource Specification – The `-xrm` option of `dxterm` that specifies a resource string to be used does not work properly.

4.4.11.3 dxterm Does Not Clear Out /etc/utmp – Do not end your login session by selecting the Quit menu item under the Session menu item through the Session Manager if there are one or more `dxterms` running. First logout out of each `dxterm`, then select the quit menu item. Failure to follow this procedure will leave `ttyXX` entries in `/etc/utmp`. Thus, users remotely logged into your workstation will see erroneous data when using commands such as `w(1)` and `who(1)`.

4.4.11.4 Using ioctl with sigio Hangs dxterm – Simultaneously using `ioctl` and the `sigio` signal will hang the `dxterm` on a workstation and will eventually hang the system such that nothing works (including any network connections). You will have to reset and reboot the system. The arguments to the `ioctl` and `fnctl` calls are as follows:

```
if((filed < 0) || (sigvec(SIGIO, &iovec, 0) == -1) ||
                (fcntl(filed, F_SETFL, FNDELAY|FASYNC) == -1 ) ||
                (ioctl(filed, FIONBUF, &_nbufcnt) == -1)) {
            return RETERR;
}
```

The program will access the tape and read data. The signal never arrives.

4.4.11.5 Using System V Shell (sh5) as Default – A DECterm window does not go away when you attempt to log out if you are using AT&T's System V shell (`sh5`) as the `/usr/etc/passwd` default shell and start a DECterm session using `dxterm -ls`.

To remove the window from the screen, iconify it or end the session.

Layered Products Notes 5

The following notes apply to these ULTRIX and ULTRIX Worksystem Software layered products:

- DECphigs
- ULTRIX Mail Connection

5.1 DECphigs

The following notes apply to the DECphigs implementation of PHIGS on DECstation/DECsystem 5000 Model 200 series processors.

5.1.1 Anti-Aliasing Modes

There are three available settings for anti-aliasing, each with a separate set of functionality and restrictions. The three modes are:

Mode 0
: No anti-aliasing is performed. No restrictions due to anti-aliasing exist when mode 0 is used.

Mode 1
: Lines are 2.5 pixels wide. Anti-aliased lines are blended from the line color to the anti-aliasing background color (color table entry 0).

  Pixels are written only when the computed pixel value is greater than the value of the existing pixel. Overlapping lines are distorted when arbitrary colors are used because the numerical value of a pixel is not necessarily a good indication of pixel brightness (for example, the values of pixels within a `PseudoColor` visual are indices into a colormap, rather than color values themselves). It is the client's responsibility to set up the colormap appropriately.

  Anti-aliasing is enabled when `HLHSR_MODE` is set to `Off`. When `HLHSR_MODE` mode is set to `ZBuffer`, anti-aliasing is suppressed.

Mode 2
: Lines are 2.5 pixels wide. Anti-aliased lines are blended from the line color to the anti-aliasing background color (color table entry 0).

  All pixels of a line are written regardless of existing pixel values.

  Anti-aliasing is enabled when `HLHSR_MODE` is set to `Off`. When `HLHSR_MODE` mode is set to `ZBuffer`, anti-aliasing is suppressed.

5.1.2 Clipped Objects

Objects located on the near clipping plane in modeling space are sometimes rendered, and sometimes clipped.

5.1.3 Polygons with Nonlinear Vertex Data

Smooth-shaded polygons with nonlinear vertex color and vertex normal data are rendered with inconsistent shading.

5.1.4 Adjacent Concave Polygons

Edges drawn along two concave polygons are not rendered correctly. Half the edge of one polygon appears blocked by the other polygon. Setting the edge flag for both polygons fixes the problem.

5.1.5 Colinear Vertices

When three colinear vertices are connected by a segment, specifying edge flags on any two of the vertices may result in the corruption of the endpoints of the segment during rendering.

5.1.6 Defining Points with Identical Coordinates

Segments (edges) connecting multiple-defined vertices are not always rendered correctly. The following situations involve multiple-defined vertices:

- Duplicated vertices in a polygon
- Edges with more than two defined vertices
- Colinear triangles (triangles with two duplicate vertices or three colinear vertices)

5.1.7 Overlapping Polygons

If two polygons overlap, and the rear polygon has edge flags set, the edges of the rear polygon may bleed through the front polygon during rendering.

5.1.8 Z-Buffering and Edges

When `HLHSR_MODE` is set to `NONE` (that is, the Z-buffer is disabled) edges are not rendered correctly.

5.1.9 Trailing Pixels of Lines

A solid line's trailing pixel is not rendered.

5.1.10 Mapping a Pattern to a Line

Patterns mapped to lines shorter than the pattern itself may cause pixel dropout when the line is rendered.

5.1.11 Graphics Primitive Clipping

Line and polygon vertices which are clipped (either by near or far clipping planes or by viewing volume boundaries) may result in pixel dropout.

5.1.12 Unimplemented PHIGS Primitives

Complex QuadMesh objects and picking of annotation pixmaps are not yet implemented.

5.1.13 Recursive Structures in PHIGS

The current implementation of PHIGS does not check application code for recursive structure networks. Any structure network which references itself is a recursive structure network.

Applications containing recursive structure networks may create self-referencing loops. Executing a self-referencing loop usually results in an error similar to the following:

```
XIO: fatal IO error 32 Broken pipe on X server
```

The error occurs immediately after the application containing the recursive structure calls the execute structure function.

5.1.14 Weighting Control Points for NURBS

Only positive weights are allowed on rational control points. Nonpositive weights will generate a `BadValue` error.

5.1.15 Pixel Dropout in Polygons and NURBS

NURBS and polygons containing lines shorter than one pixel are subject to pixel dropout.

5.1.16 Tessellating a NURBS into Polygons

The upper limit on the number of polygons into which a nurb can be tessellated when using the implementation-dependent NURB approximation type is 256×256.

5.1.17 Knot Vectors in a NURBS

Knot vectors for surfaces, curves, and trim curves are subject to the following two restrictions (violations will cause a protocol error):

1. Knot multiplicities at either end of the knot vector must not be greater than the order of the curve or surface.
2. Knot multiplicities not at the ends of the knot vector must not be greater than the order minus one.

Consider, for example, a curve of order three with five control points. The protocol requires that there be exactly eight knots in the knot vector.

The following vectors violate the first restriction:

[0, 0, 0, 0, 1, 2, 2, 2] (too many instances of ‘‘0’’)
[0, 0, 1, 1, 2, 2, 2, 2] (too many instances of ‘‘2’’)

The following vectors violate the second restriction:

[0, 0, 0, 1, 1, 1, 2, 2] (too many instances of ‘‘1’’)
[0, 1, 2, 2, 2, 2, 3, 4] (too many instances of ‘‘2’’)

5.1.18 Supported Color Approximation Types

The following table lists the color approximation types available using different visual types on 8- and 24-plane DECstation/DECsystem 5000 Model 200 series processors. A legend follows the table.

Visual Type	24-Plane Display	8-Plane Display
DirectColor	CS & CR	n/a
TrueColor	CS & CR-Limited	CS-Flat & CR-Limited
PseudoColor	n/a	CS-Flat & CR-100
StaticColor	n/a	CS-Flat & CR-100 (Predef)
GreyScale	n/a	CS-Flat & CR-100
StaticGrey	n/a	CS-Flat & CR-100 (Predef)

n/a	Visual type is not available with the specified display depth.
CS	ColorSpace is supported with no special caveats.
CS-Flat	ColorSpace with flat shading is supported. Interpolating between colors in the color space results in undefined values.
CR	ColorRange is supported with no special caveats.
CR-Limited	ColorRange is supported in a limited way. A TrueColor visual allows only 3 possible predefined color ranges, one along each color axis. TrueColor will work only if the display depth permits an equal number of samples for each color component. For example, 256 samples for 8/8/8 (a 24-plane display) is acceptable, but 8 and 4 for 3/3/2 (an 8-plane display) is not.
CR-100	ColorRange is supported but the three multipliers (mults) used to pack the color components into a single 32-bit pixel are forced to (1,0,0) to conform to PHIGS Version 4.0P semantics. In PHIGS Version 5.0, the client is responsible for setting the mults to useful values.
Predef	A reasonable ColorSpace or ColorRange encoding must already exist in a predefined color map.

5.1.19 Using a ColorRange

Due to round-off algorithms, the first and last entries in a multi-entry ColorRange are not consistently generated. For a continuously changing ColorRange, this is not noticeable. For a static ColorRange that has discontinuities at the start or finish of

the range, at least one entry at each end should be used to ''pad'' the ColorRange and thus ensure its integrity.

5.1.20 Structure Storage Limit

It is not possible to determine exactly how many structures, structure elements, or both can be created in the PHIGS central structure store. This is because the maximum size of the central structure store is dependent upon the following:

- The memory available to the server process (note that the memory allocated to the server and other processes cannot exceed the swap space for the workstation).
- The type of data which is being stored.

The memory requirements for structures and structure elements are given in the following table. The memory requirements are given in bytes and can be used to obtain an upper bound for the number of structures and structure elements that can be created.

Data Type	Memory Requirement in Bytes
Structure	85
Label element	16 + `sizeof(PHIGSLabelInfo)`
Execute element	32 + `sizeof(PHIGSExecuteStructure)`
Primitive element	32 + 2 × `sizeof(PHIGSXXX)`
Other elements	8 + `sizeof(PHIGSXXX)`

5.2 ULTRIX Mail Connection

The following notes apply to the ULTRIX Mail Connection.

5.2.1 Installing ULTRIX Mail Connection Version 1.1 on ULTRIX/UWS Version 4.1

Because of changes in `setld`, you will need to set an environment variable before attempting to install ULTRIX Mail Connection (UMC) Version 1.1.

To install UMC Version 1.1, follow these steps:

1. Read the `setld(8)` note in the section on Superuser Commands in Chapter 3 and follow the instructions there.
2. If you have installed the base system version of `MH` (Message Handler), you must delete it before installing UMC Version 1.1.

To delete the base system version of `MH` on ULTRIX/UWS Version 4.1 (VAX), enter this command:

```
# setld -d ULTMH410
```

To delete the base system version of `MH` on ULTRIX/UWS Version 4.1 (RISC), enter this command:

```
# setld -d UDTMH410
```

The UMC version of MH is installed in a different location from the base system version. We recommend that you create symbolic links from the old location to the new location so that existing MHusers do not need to take any action. To create the symbolic links for VAX or RISC, enter the following commands:

```
# ln -s ../bin/mh /usr/new/mh
# ln -s ../../lib/mh /usr/new/lib/mh
```

3. Install UMC Version 1.1 according to the instructions in the *ULTRIX Mail Connection Installation Guide*.

Documentation Notes 6

The following notes apply to ULTRIX, ULTRIX Worksystem Software, and layered products documentation

6.1 ULTRIX Documentation

The following notes apply to ULTRIX documentation and cover the following topics:

- Installation
- Software Development
- Network and Communications
- Security
- POSIX and XPG
- ULTRIX/SQL
- Reference Pages

6.1.1 Installation

The following notes apply to the installation of ULTRIX/UWS Version 4.1.

6.1.1.1 Installation Guides and Product Authorization Keys (PAKs) – The *Advanced Installation Guide* and the *Basic Installation Guide* indicate that you cannot bring your system to multi-user mode without first registering a Product Authorization Key (PAK). This is incorrect. Each system has a two-user license as the default. If you want to increase your system's simultaneous login capacity to more than two users, you must register a PAK.

6.1.1.2 Creating Copies of Sparse Dump Files – A sparse file utility is documented in the *Guide to System Crash Recovery*. This section describes how to create a permanent copy of crash dump files on tape.

To create a permanent copy of the dump files on tape, use the `tar` command to extract the dump files. You should compress the dump files before copying them to tape because the `vmcore` file created by savecore is a sparse file which will expand when you copy it to tape. To compress the dump files, use the `compress` command. To copy the dump files to tape, type the following command sequence:

```
# compress path/vmunix.n path/vmcore.n
# tar c path/vmunix.n.Z path/vmcore.n.Z
```

The path is the directory pathname specified in the /etc/rc.local file such as /usr/adm/crash. The time a system crash occurs, n is incremented by one. For example, if path is /usr/adm/crash and n is 1, type the following command sequence:

```
#  compress /usr/adm/crash/vmunix.1 /usr/adm/crash/vmcore.1
#  tar c /usr/adm/crash/vmunix.1.Z /usr/adm/crash/vmcore.1.Z
```

After you specify the tar command, use the rm command to remove the dump files and to conserve space on the specified file system. The following example shows how to remove the dump files. In this example the dump files are located in /usr/adm/crash and n is 1.

```
#  rm /usr/adm/crash/vmunix.1.Z /usr/adm/crash/vmcore.1.Z
```

To decompress the dump files when extracting them from a tape, use the cat, uncompress, and dd commands. The dd command has an option to create sparse output files. Remember that the vmcore file created by savecore is a sparse file. To extract and decompress the dump files from tape, type the following command sequence:

```
# tar x
x path/vmunix.1.Z, n bytes, n blocks
x path/vmcore.1.Z, n bytes, n blocks
# cat path/vmunix.1.Z | uncompress | dd conv=sparse of=path/vmunix.1
# cat path/vmcore.1.Z | uncompress | dd conv=sparse of=path/vmcore.1
# rm path/vmunix.1.Z path/vmcore.1.Z
```

To create a copy of the dump files on another system or in another directory, use the dd command to copy the files. Remember that the vmcore file created by savecore is a sparse file. If you simply copy this file, using the cp command for instance, it will expand and use up much file system space. Thus, to reserve file system space, you can copy the sparse files using the dd command. To copy the dump files to another directory using the dd command, log in as root or become superuser and enter the following two commands using this syntax:

dd conv=sparse if=*path*/vmunix.1 of=*newpath*/vmunix.1
dd conv=sparse if=*path*/vmcore.1 of=*newpath*/vmcore.1

The *path* is the directory pathname specified in the /etc/rc.local file such as /usr/adm/crash. The *newpath* is the directory pathname where you want to copy the dump files.

To copy the dump files to another system using the dd command, type the following command sequence:

```
# compress path/vmunix.1 path/vmcore.1
# rcp path/vmunix.1.Z path/vmcore.1.Z system:/usr/savecrash
# rlogin system
# cd /usr/savecrash
# cat vmunix.1.Z | uncompress | dd conv=sparse of=vmunix.1
# cat vmcore.1.Z | uncompress | dd conv=sparse of=vmcore.1
# rm vmunix.1.Z vmcore.1.Z
```

6.1.1.3 Estimating Disk Space for Partial Crash Dumps – In the *Advanced Installation Guide* and in the *Guide to System Crash Recovery,* tables similar to the following provide guidelines on the amount of disk space to allocate for partial crash dumps. However, neither book addresses how these numbers are determined.

Examples of Partial Crash Dump Space

Physical Memory	Maximum Number of Users	Space Needed for RISC	Space Needed for VAX
6 megabytes	2	NA	6 megabytes
8 megabytes	2	8 megabytes	8 megabytes
16 megabytes	16	14 megabytes	10 megabytes
32 megabytes	32	20 megabytes	12 megabytes
64 megabytes	64	28 megabytes	16 megabytes
128 megabytes	128	40 megabytes	26 megabytes
256 megabytes	128	40 megabytes	26 megabytes
512 megabytes	256	48 megabytes	34 megabytes

In determining the crash dump space, the following factors are accounted for:

- The size of the kernel (KERNEL)

  The KERNEL is estimated at 4 MB for both RISC and VAX.

- The size of the kernel memory allocator (KMALLOC DATA)

  KMALLOC DATA is calculated by using the following formulas:

  - RISC:

    This value is an estimate based on the physical memory size (PHYSMEM). However, the maximum KMALLOC DATA sized is fixed at 16 MB no matter what the physical memory size.

  - VAX:

    KMALLOC DATA is 3 MB if PHYSMEM is less than 40 MB.

    KMALLOC DATA is 10 MB if PHYSMEM is greater than or equal to 100 MB

    KMALLOC DATA is PHYSMEM divided by 10 MB for other PHYSMEM sizes.

- The size of the user areas (UAREAS)

  The UAREAS is calculated by the following formulas:

  - RISC:

    ((20 + 8 * MAXUSERS) * UPAGES * NBPG)

  - VAX:

    ((20 + 8 * MAXUSERS) * UPAGES * NBPG)

- The size of the page tables (PAGETBLS)

The PAGETBLS is calculated by the following formulas:

- RISC:

  ((3 * 8 * MAXUSERS + 20 * 3) * NBPG)

  This value is an estimate based on the physical memory size. However, the system allows a maximum of 12 MB for partial crash dumps of PAGE TABLES regardless of physical memory size. The formula given is not used because it gives gross page table sizes. The actual page table size should never be as large as the formula indicates.

- VAX:

  If PHYSMEM is less than 64 MB then (32 * NUMPTEPG * NBPG) or 2 MB
  If PHYSMEM greater than 64 MB then (64 * NUMPTEPG * NBPG) or 4 MB

  Allow at most 4 MB for dumping PAGE TABLES regardless of physical memory size.

  Maximum user virtual address space (MAXUVA) can be configured in the system configuration file to get more PAGE TABLE space in memory. This would require an adjustment in the calculation.

To determine the size of each factor, you can use the following constants in your calculations:

Constant	RISC	VAX
Number of pages used for the user areas for each active/inactive processes in the system (UPAGES)	2	14
Number of page table entries for each page (NUMPTEPG)	1024	128
Number of bytes for each page (NBPG)	4096	512

The following chart shows the resultant calculation of each of the previously mentioned factors for a given configuration:

```
+-------------------------------------------------------------------------+
|                          |                          |                   |
|  16 MB / 16 MAXUSERS     |  32 MB / 32 MAXUSERS     |  64 MB / 64 MAXUSERS  |
|                          |                          |                   |
|--------------------------|--------------------------|-------------------|
|  KERNEL      4     4     |  KERNEL    4     4       |  KERNEL     4     4   |
|  KMALLOC     3     5     |  KMALLOC   4     8       |  KMALLOC    6    10   |
|  UAREAS      1     1     |  UAREAS    2     2       |  UAREAS     4     4   |
|  PAGETBLS    2     4     |  PAGETBLS  2     6       |  PAGETBLS   2    10   |
|             ---  ----    |            ---   ---     |             ---  ----   |
|  TOTALS     10MB  14MB    |  TOTALS   12MB  20MB     |  TOTALS    16MB  28MB  |
|-------------------------------------------------------------------------|
|                          |                          |                   |
| 128 MB / 128 MAXUSERS    | 256 MB / 128 MAXUSERS    | 512 MB / 256 MAXUSERS |
|                          |                          |                   |
|--------------------------|--------------------------|-------------------|
|  KERNEL      4     4     |  KERNEL    4     4       |  KERNEL     4     4   |
|  KMALLOC    10    16     |  KMALLOC  10    16       |  KMALLOC   10    16   |
|  UAREAS      8     8     |  UAREAS    8     8       |  UAREAS    16    16   |
|  PAGETBLS    4    12     |  PAGETBLS  4    12       |  PAGETBLS   4    12   |
|             ---  ----    |            ---   ---     |             ---  ----   |
|  TOTALS     26MB  40MB    |  TOTALS   26MB  40MB     |  TOTALS    34MB  48MB  |
+-------------------------------------------------------------------------+
```

6.1.1.4 Guide to Diskless Management Services – The following notes apply to the *Guide to Diskless Management Services*.

6.1.1.4.1 Subset Sizes – The subset sizes listed in Section 2.3.2.1 of the *Guide to Diskless Management Services* have been updated for this release in Table 6-1.

Table 6-1: Approximate Disk Space Required

Software Subsets	Approximate Size in Mbytes (RISC)	Approximate Size in Mbytes (VAX)
All ULTRIX/UWS	263	141
All ULTRIX/UWS unsupported	39	42

Based on these revised figures, the sample calculation of disk space required for a `dlenv` file system to be accessed by RISC clients is updated as follows:

```
All ULTRIX                    263 Mbytes
All ULTRIX unsupported         36 Mbytes
One layered product            50 Mbytes

Subtotal                      342 Mbytes

20% file system allowance      68 Mbytes

Total                         410 Mbytes
```

6.1.1.4.2 Boot Command for DECstation/DECsystem 5000 Model 200 – The boot command is not correct for DECstation/DECsystem 5000 Model 200 processors. Please refer to the section on the DECstation/DECsystem 5000 Model 200 Series in "Processor–Specific Notes" in this document for the correct syntax.

6.1.2 Software Development

The following notes apply to documentation about software development.

6.1.2.1 Additions to the Kernel Messages Manual – The following are new panic messages that are not included in the *Kernel Messages Manual:*

Message: `panic: sm_clear_dev_tlbs: miss proc-to-shm pointer`

Output: none

File: `sys/vm/mips/sm_machdep.c`

Routine: `sm_clear_dev_tlbs`

Description: A process does not point back to a shared memory segment which points to the process.

User Action: File a problem report.

Message: `panic: SO_LOCK: s->ref hung`

Output: none

File: `sys/h/socketvar.h`

Routine: Many, the message is a macro.

Description: The message can happen only on a single-processor system. It indicates a design problem in the SMP locking routines in the network subsystem.

User Action: File a problem report.

6.1.3 Networking and Communications

This section contains documentation about networking and communications.

6.1.3.1 Corrections to the Guide to Kerberos – The *Guide to Kerberos* states, on page 4-13, that the `bindsetup` command adds the following lines to `/etc/rc.local:`

```
# %BINDSTART - BIND daemon
[ -f /usr/etc/named ] && {
/usr/etc/named -s -a kerberos one -b /var/dss/namedb/named.boot;
     echo -n ' named' >/dev/console
}
```

This is incorrect. The bindsetup command actually adds the following lines:

```
# %BINDSTART - BIND daemon
[ -f /usr/etc/named ] && {
  /usr/etc/named -n -a kerberos.one -b /var/dss/namedb/named.boot;
     echo -n ' named' >/dev/console
}
```

In Section 4.6, Changing the Master Key of the Kerberos Database, step 3 in the numbered procedure should be placed after step 5.

In Section 4.5, Starting the Kerberos-Authenticated named Daemon, the following line is incorrect:

```
/usr/etc/named -s -a kerberos one -b /var/dss/namedb/named.boot;
```

This line should read as follows:

```
/usr/etc/named -n -a kerberos.one -b /var/dss/namedb/named.boot;
```

6.1.3.2 Correction to Root Name Server Reference – The *Guide to the BIND/Hesiod Service* contains an incorrect reference to the host name and address of the root name server.

As of April 1990, the root name server, nic.ddn.mil with IP address 26.0.0.73, is on ns.nic.ddn.mil with an IP address 192.67.67.53. Please observe these corrections when referring to the *Guide to the BIND/Hesiod Service* .

6.1.3.3 Documentation for DEMNA XNA Interface – The *Guide to the Data Link Interface* does not include the new DEMNA XNA interface in the list of supported hardware.

6.1.3.4 Corrections to Guide to Preparing Software for Distribution on ULTRIX Systems and the kitcap(5) Reference Page – There is incorrect syntax for `/etc/kitcap` file entries and for `gentapes` and `genra` command lines in `kitcap`(5) and in the *Guide to Preparing Software for Distribution on ULTRIX Systems*.

The corrections follow.

6.1.3.4.1 Section 5.8, Building /etc/kitcap – The syntax and examples in Section 5.8, Building /etc/kitcap, are incorrect. The correct syntax and examples follow.

The format of an `/etc/kitcap` entry for tape media is:

```
<product code><media code> | [product description]:<directory1>[:directory2]:\
SPACE:SPACE:SPACE:INSTCRTL:<subset1>[:<subset2>]
```

The subsets must be listed so that any subset on which other subsets depend is listed before its dependent subsets.

The example that follows shows an entry for TK50 tape media:

```
UWS400TK | ULTRIX Worksystem Software:/sys/dist/:\
SPACE:SPACE:SPACE::INSTCTRL:UWSXRT400:UWSMH400
```

The format of an `/etc/kitcap` entry for disk media is:

```
<product_code><media_code>:partition:dd=<destination_directory>:\
[product_description]:<directory1>[:directory2]:instctrl:<subset1>[:<subset2>]
```

The example that follows shows an entry for RA60 disk media:

```
UWS400RA:c:dd=product:ULTRIX_Worksystem_Software:/sys/dist/:\
instctrl:UWSXRT400:UWSMH400
```

The underscore character (_) is required to connect words in a product description for disk media. The subsets must be listed so that any subset on which other subsets depend is listed before its dependent subsets.

6.1.3.4.2 Section 5.9.1, Making Tape Media – The syntax and example in Section 5.9.1, Making Tape Media, are incorrect. The correct syntax and examples follow.

Use the `gentapes` utility to make tape media. The command line syntax is:

```
gentapes [-wv] [hostname:]<product_code> <special>
```

The -w option indicates write only; the -v option indicates verify only. If neither option is specified, the utility writes, rewinds the tape, then verifies.

If you specify a `node,` the `gentapes` utility looks for the output directory on the node you specify. For example:

```
# gentapes mysystem:UWS400 /dev/nrmt0h
```

You can use the Network File System (NFS) to remotely mount the kit on a machine with the correct drive.

The `gentapes` utility appends either TK or MT to the `product_code` after finding the entry in the `/etc/kitcap` file. For example, if you type the following command and the `/etc/kitcap` entry specifies TK50 tape, the `gentapes` utility appends TK to UWS400:

```
# gentapes UWS400 /dev/nrmt0h
```

6.1.3.4.3 Section 5.9.2, Making RA60 Disk Media – The syntax and example in Section 5.9.2, Making RA60 Disk Media, are incorrect. The correct syntax and an example follow.

Use the `genra` utility to make RA60 disk media. The command line syntax is:

```
genra [-wv] [hostname:]<product_code> <special>
```

The following example specifies a `hostname:`

```
genra mysystem:UWS400 /mnt
```

The `genra` utility appends RA to the `product_code` automatically after finding the entry in the `/etc/kitcap` file.

6.1.4 Security

The following notes apply to documentation about security issues.

6.1.4.1 Incorrect Subset in Security Guide for Administrators – In the *Security Guide for Administrators,* Chapter 7, Starting and Configuring a Secure System, incorrectly states that ULTSEC040 is the name of the ULTRIX security software. The correct name is ULTSEC410.

All occurrences of ULTSEC040 should read ULTSEC410.

6.1.4.2 Controlling Network Access to Workstation Displays – Chapter 6 of the *Security Guide for Users* incorrectly states that if there are differences between the system access control list and a workstation access control list, the system access control list prevails. This is not true.

The correct documentation is:

At system startup, the X server initializes the server access control list by reading the `/etc/X*.hosts` file. This privileged file names the hosts on a network that can access a workstation display.

When the Session Manager (`dxsession`) is started, it updates this server access control list to match the session access control list. The session access control list is the list of hosts that users specify using the Security... option from the Customize menu from the Session Manager window. The Session Manager stores this list of hosts in the `.Xdefaults` file in the user's home directory, using the resources `sm.host_list` and `sm.num_hosts`.

The session access control list, if it exists, overrides the server access control list. For example, if the server access control list includes hosts `orion` and `myrtle`, and the session access control list includes only host `myrtle`, the Session Manager requests that the server remove `orion` from the server access control list, and only authorized users on `myrtle` can access the workstation.

If a user does not add a host to the session access control list using the Security... option from the Customize menu, or if the user does not save the changes made during the current session, the Session Manager does not create a list of entries for the `sm.host_list` and `sm.num_hosts` resources in the `.Xdefaults` file. Thus, no session access control list exists. If no session access control list exists, the only hosts allowed access to the workstation display are those listed in the server access control list before the session is initiated. These are the hosts listed in the `/etc/X*.hosts` file.

6.1.5 POSIX and XPG

The following notes apply to standards and external specifications, specifically the documentation in the *POSIX Conformance Document*.

6.1.5.1 The cpio Command – In running the X/OPEN Verification Suite, Release 3.203, in POSIX mode, the following exception was found in the `cpio` command.

The ULTRIX `cpio` command properly creates and extracts POSIX conformant files. However, the *MAGIC* number in the POSIX `cpio` header file is an integer, not an ASCII string. This will only affect applications that utilize the header file. There is no effect on reading and writing of POSIX `cpio` archives.

The workarounds are as follows:

- Modify the header file, `/usr/include/cpio.h` so that the *MAGIC* number is in quotation marks.
- Convert the integer to an ASCII string at runtime.

6.1.5.2 The tcsendbreak Library Call – In running the X/OPEN Verification Suite, Release 3.203 in POSIX mode, the following exception was found in the `tcsendbreak` library call.

Due to a hardware limitation on DECstation 2100s and 3100s, the calling of `tcsendbreak` to transmit a break condition, generates an extra null character following a sequence of zero-value bits that continues for more than the time required to send one byte.

6.1.5.3 The tar Command – In running the X/OPEN Verification Suite, Release 3.203, in POSIX mode, the following exceptions were found in the `tar` command:

- Prefix Usage and File Names of 100 to 256 Characters
- Permissions
- Multiple volumes

6.1.5.3.1 Prefix usage and file names of 100 to 256 characters – According to Chapter 10, Section 10.1.1 of IEEE Std 1003.1-1988, "The *name* and the *prefix* fields produce the pathname of the file. The hierarchical relationship of the file is retained by specifying the pathname as a path prefix, a slash character and filename as the suffix. If the *prefix* , contains non-null characters, *prefix* , a slash character, and *name* are concatenated without modification or addition of new characters to produce a new pathname."* The calculation used by ULTRIX `tar` is *prefix* and *name* concatenated; ULTRIX does not use the slash (/). In ULTRIX, if the *prefix* is null, the pathname is *name* as in POSIX.1.

The *prefix* field has a backward overflow for file names greater than 100 characters. The calculation used when an overflow occurs is as follows:

```
filename: /a01/a02/a03/ ... /a99
```

POSIX:

```
name:   a74/a75/ ... /a99
prefix: /a01/a02/ ... /a75
```

ULTRIX:

```
name:   a99
prefix: /a01/a02/ ... /a98/
```

To work around this problem, use file names with less than 100 characters.

6.1.5.3.2 Permissions – According to Chapter 10, Section 10.1.1 of IEEE Std 1003.1-1988, a process with appropriate privileges restores the ownership and permissions exactly as recorded on the medium, except that symbolic user and group IDs are used for the `tar` format. If only the uppercase `–P` option to the `tar` command is used, ULTRIX does not restore permissions as they were preserved on the media. The lowercase `–p` option to the `tar` command allows the modes to be preserved, and also allows non-permitting processes to preserve the modes.

Barring the restrictions previously noted, using `tar` with both the uppercase `–P` and lowercase `–p` options (`tar –Pp`) insures that permissions are preserved.

---

* *IEEE Standard Portable Operating System Interface for Computer Environments* (New York, NY: The Institute of Electrical and Electronics Engineers, Inc, 1988)

6.1.5.3.3 Multiple Volumes – The `tar` command does not conform to Chapter 10, Section 10.1.3 of IEEE Std 1003.1-1988 in regard to multiple volumes. The `tar` command supports multiple volumes in non-POSIX mode with no restrictions. However, when used in POSIX mode, any file that spans two media is corrupted. All other files are preserved.

To work around the problem, insure that all files are contained fully in a single medium.

6.1.6 ULTRIX/SQL

The following note applies to ULTRIX/SQL.

6.1.6.1 VAX Kernel Configuration Parameter Specified Incorrectly in ULTRIX/SQL Operations Guide – In Section 3.3.4.1, Shared Memory -- VAX Systems, of the *ULTRIX/SQL Operations Guide*, the kernel configuration parameter **smsmat** is incorrectly listed as **smsat** in two places. Note that you need to specify the **smsmat** parameter only for VAX systems with fewer than 32 megabytes of internal memory.

6.1.7 Reference Pages

The following notes apply to the reference pages.

6.1.7.1 New and Changed Reference Pages – The following reference pages are new or have been changed in this release:

```
chroot.1

mt.1

mblen.3int

mbstowcs.3int

mbtowc.3int

rrpc_are_you_there.3ncs

wctomb.3int

mdc.4

mti0.4

ne.4

presto.4

rz.4

scsi.4

tz.4
```

```
dupterm.8

kgconv.8

presto.8

prestoctl_svc.8

scamp.8

rzdisk.8

dxpresto.8
```

6.1.7.2 Reference Pages Available Only Online – For ULTRIX/UWS Version 4.1 the new and changed reference pages are available only online.

If you need a copy of any of the reference pages on paper, as opposed to online, you can process the source file for the reference page and print the formatted page.

The source files for all the reference pages are stored in subdirectories of the `/usr/man` directory after you install your ULTRIX system. For example, the `/usr/man/man4` directory contains source files for Section 4 reference pages. Each source file in that directory contains one reference page. The names of the source files indicate which reference page they contain. For example, the `rz`(4) reference page is stored in the `rz.4` file. The `scsi`(4) reference page is stored in the `scsi.4` file, and so on.

To process a reference page source file, use the `tbl` preprocessor. Process the output from `tbl` using the `nroff` command with the `-man` macro. You can print the output from the `nroff` command on a line printer. For example, to process the `rz.4` source file using `nroff`, issue the following command:

```
% tbl /usr/man/man4/rz  | nroff -man | col | lpr &
```

6.2 ULTRIX Worksystem Software Documentation

The following notes apply to ULTRIX Worksystem Software documentation and cover the following topics:

- Online Software Product Description (SPD)
- Xlib Manual Additions
- Macros Described in Appendix C of the Xlib Manual
- Discrepancies between DECwindows Toolkit and the Toolkit documentation
- Reference Pages

6.2.1 Online Software Product Description (SPD)

An online copy of the UWS Software Product Description is provided as a reference to the software and hardware configurations that ULTRIX/UWS Version 4.1 supports. The online SPD is located in the `/usr/etc` directory. The online SPD filename is `spd_uws`.

Note

This electronic copy of the SPD is to be used only as a guide, and is not warranted to be accurate or complete, nor is it to be used as a substitute for the printed SPD that comes with your software. The printed SPD is the legal document listing supported software components and supported hardware configurations for your distribution.

6.2.2 Xlib Manual Additions

The following functions were added to the *Guide to the Xlib Library* in the ULTRIX/UWS Version 2.2 release.

6.2.2.1 XVisualIDFromVisual – To obtain the visual ID from a `Visual`, use `XVisualIDFromVisual`.

```
VisualID XVisualIDFromVisual(visual)
        Visual *visual;
```

visual Specifies the visual type.

The `XVisualIDFromVisual` function returns the visual ID for the specified visual type.

6.2.2.2 XDisplayKeyCodes – To obtain the legal KeyCodes for a display, use `XDisplayKeycodes`.

```
XDisplayKeycodes(display, min_keycodes_return,
max_keycodes_return)
          Display *display;
          int *min_keycodes_return, max_keycodes_return;
```

display Specifies the connection to the X server.

min_keycodes_return
Returns the minimum number of KeyCodes.

max_keycodes_return
Returns the maximum number of KeyCodes. The `XDisplayKeycodes` function returns the minimum number of keycodes and maximum number of KeyCodes supported by the specified display. The minimum number of KeyCodes returned is never less than 8, and the maximum number of Keycodes returned is never greater than 255. Not all KeyCodes in this range are required to have corresponding keys.

6.2.2.3 XResourceManagerString – To obtain a pointer to the resource manager string of a display, use `XResourceManagerString.`

```
char *XResourceManagerString(display)
        Display *display;
```

display Specifies the connection to the X server.

The `XResourceManagerString` returns the RESOURCE_MANAGER property from the server's root window of screen zero, which was returned when the connection was opened using `XOpenDisplay`.

6.2.2.4 XAddExtension – The `XAddExtension` function was added in the ULTRIX/UWS Version 2.2 release to the *Guide to the Xlib Library*, Appendix C:

```
XExtCodes *XAddExtension(display)
        Display *display;
```

For local Xlib extensions, `XAddExtension` allocates the `XExtCodes` structure, bumps the extension number count, and chains the extension onto the extension list.

To transmit variable length data, use the `Data` macros. If the data fits into the output buffer, then this macro copies it to the buffer. If it does not fit, however, the `Data` macro calls `_XSend`, which transmits first the contents of the buffer and then your data. The `Data` macros take three arguments: the Display, a pointer to the beginning of the data, and the number of bytes to be sent.

```
Data(display, (char *) data, nbytes);

Data16(display, (short *) data, nbytes);

Data32(display, (long *) data, nbytes);
```

`Data`, `Data16`, and `Data32` can use their last argument more than once, so that argument should be a variable rather than an expression such as ''nitems*sizeof(item)''. This kind of computation should be done in a separate statement before calling the macros.

Use the appropriate macro when sending byte, short, or long data.

6.2.2.5 XRead Functions – If there is variable length data after the reply, change the `True` to `False`, and use the appropriate `_XRead` function to read the variable length data.

`_XRead` reads the specified number of bytes into data.

```
_XRead(display, data, nbytes)
       Display *display;
       char *data;
       long nbytes;
```

`_XRead16` reads the specified number of bytes, unpacking them as 16-bit quantities, into the specified array as shorts.

```
_XRead16(display, data, nbytes)
        Display *display;
        short *data;
        long nbytes;
```

_XRead32 reads the specified number of bytes, unpacking them as 32-bit quantities, into the specified array as longs.

```
_XRead32(display, data, nbytes)
        Display *display;
        long *data;
        long nbytes;
```

_XRead16Pad reads the specified number of bytes, unpacking them as 16-bit quantities, into the specified array as shorts. If the number of bytes is not a multiple of four, _XRead16Pad reads up to three additional pad bytes.

```
_XRead16Pad(display, data, nbytes)
        Display *display;
        short *data;
        long nbytes;
```

_XReadPad reads the specified number of bytes into data. If the number of bytes is not a multiple of four, _XReadPad reads up to three additional pad bytes.

```
_XReadPad(display, data, nbytes)
        Display *display;
        char *data;
        long nbytes;
```

6.2.2.6 XLookupString – Unlike most Xlib functions which return a string, XLookupString does not return a null-terminated string. Instead XLookupString returns a count of the characters in the string. This feature is not clearly described in the Xlib documentation.

6.2.3 Discrepancies Between DECwindows Toolkit and the Toolkit Documentation

This section describes inaccuracies and omissions in the Toolkit documentation.

6.2.3.1 XtRegisterClass – The DRM Register Class routine XtRegisterClass is documented to have the widget class parameter passed by reference. The code currently requires it to be passed by value.

6.2.3.2 XtDisplayInitialize – The Display Initialize routine XtDisplayInitialize is incorrectly documented as showing the parameter display_name to be a character string passed by descriptor. It should be documented as a Display structure pointer passed by value.

6.2.4 XUI Toolkit Manual

On page 10-7 in the XUI Toolkit, Programming Volume 2 there is an error in the definition of the second argument of `DwtGetNextSegment(...)`. It is incorrectly listed as:

```
char *text_return
```

It should be:

```
char **text_return
```

6.2.4.1 DwtGetNextSegment Function – The parameters to the compound string utility `DwtGetNextSegment` routine discussed in the UWS documentation are incorrect. The `direction_r_to_l` parameter that is shown in the *XUI Toolkit Reference Manual* as the address of a Boolean should be the address of an integer. Passing the address of a Boolean to this routine yields unexpected results, including alignment errors on some platforms.

6.2.5 UID File Descriptions

There are no descriptions of the files in the `/usr/lib/X11/uid/` directory. The files in this `uid` directory are compiled versions of the User Interface Language (UIL) files that define the form (menus, buttons, command callbacks, fonts, geometry, and so on) for the functionality of the applications.

Additionally, the *UWS Reference Pages* for various applications do not mention the UID files, even though some other files related to the applications, in particular the app-defaults files, are shown.

6.2.6 Reference Pages

This section discusses changes to the *UWS Reference Pages* since ULTRIX/UWS Version 4.0.

6.2.6.1 X Server Reference Pages – Section 8 reference pages based on the VAX and RISC versions of `X(8X)` have been added to the UWS reference pages inventory since the UWS Version 2.1 release. The reference pages are as follows:

- `Xmfb(8X)` - RISC version with a cross-reference from `Xcfb`.
- `Xqvsm(8X)` - VAX version with cross-references from `Xqdsg` and `Xgb`.
- `Xtm(8X)` - RISC version for DECstation/DECsystem 5000 Model 200PXG and PXG-turbo series processors.
- `Xtm2d(8X)` - RISC version for DECstation/DECsystem 5000 Model 200PX series processors.

6.2.6.2 DwtMainWindow Reference Page (DwtNcolormap Attribute) – The `DwtMainWindow` reference page lists the inherited attribute `DwtNcolormap`. User-created colormaps cannot replace the `DwtNcolormap` attribute. Any attempt to use a user created colormap are ignored. This is not explicitly stated in the reference page.

6.3 Layered Products Documentation

The following notes apply to layered products documentation.

6.3.1 Correction to Encryption Upgrade Installation Instructions

The *Encryption Upgrade Installation Instructions* document is in error. Some commands and libraries have been added to the Encryption Upgrade Installation kit and the information for the CDROM media is incorrect.

The correct lists of commands and libraries follow.

The encryption upgrade kit adds these commands to your ULTRIX operating system:

```
/usr/bin/crypt
/usr/bin/enroll
/usr/bin/secretmail
/usr/bin/xsend
/usr/bin/xget
```

The encryption upgrade kit replaces these libraries:

```
/usr/lib/libc.a
/usr/lib/libcg.a  (VAX only)
/usr/lib/libcP.a
/usr/lib/libcPg.a  (VAX only)
/usr/lib/libc_p.a  (VAX only)
/usr/lib/libcP_p.a  (VAX only)
```

The correct entry for CDROM media in Table 1, Encryption Upgrade Distribution Media and Device Special File or Mount Point, follows:

Distribution Media and Label	Device Special File or Mount Point
1 CDROM optical disc ULTRIX/UWS Version ULTRIX/UWS Version 4.1 ENCRYPTION (RISC/VAX)	`/mnt/RISC/ENCRYPTION` (RISC processors) `/mnt/VAX/ENCRYPTION` (VAX processors)

Problems Resolved Since Last Release A

This appendix discusses the software and documentation problems with the ULTRIX operating system and ULTRIX Worksystem Software that have been resolved since the last release of the ULTRIX operating system.

A.1 ULTRIX Problems Resolved Since Last Release

This section discusses problems in the ULTRIX operating system that have been resolved since the last release. Table A-1 lists the topic or the name of the component that has been resolved, a definition of the problem that has been resolved, and, when applicable, a reference to a Software Performance Report (SPR) or Customer Level Distribution (CLD).

Table A-1: ULTRIX Problems Resolved Since Last Release

Component	Problem Resolved	CLD/SPR
`atexit`	The `atexit` routine could reference unallocated memory.	–
`BIND/Hesiod hesupd`	The problem described in ULTRIX Version 4.0 release note 18, BIND/Hesiod Password Causes hesupd Failure, has been resolved. Now the distributed password file no longer becomes corrupted when users with a null password field attempt to change their password.	–
`BIND/Hesiod named`	The Kerberos authenticated BIND/Hesiod `named` produced many zone transfer errors in `syslog` due to timing problems.	–
`BIND/Hesiod named`	BIND/Hesiod `named` arguments were incomplete in a `ps` listing.	–
`CI Network Subsystem`	In ULTRIX Version 4.0, the performance of the CI network subsystem on DECsystem 5800s was less than the Ethernet throughput. This problem has been resolved. Now the CI network throughput on DECsystem 5800s is anywhere from one to two times faster than the Ethernet throughput.	–
`Configuration File`	The kernel would not boot if the `QUOTA` option was not included in the system configuration file (RISC processors only).	–

Table A-1: (continued)

Component	Problem Resolved	CLD/SPR
curses(3x)	The problem described in ULTRIX Version 4.0 release note 2.13.3.2, The addch() Function Causes Text Scrolling Problems, has been resolved. If your application links in the BSD curses libraries libcurses.a or libcurses_p.a (-lcurses, or -lcurses_p) you need to relink your application in order to take advantage of this fix.	SPR ICA-20807
dtoa	The problem described in ULTRIX Version 4.0 release note 2.13.10, A printf Problem (RISC only), has been resolved. The %f format (C) and F format (FORTRAN) rounded incorrectly. The fcvt function also rounded incorrectly.	–
GNUemacs	The startup of GNUemacs resulted in a segmentation fault.	CLD IPO3736
ftp	The ftp "macdef" token was unrecognized.	SPR 704
gno_close()	The kernel routine copen() was modified to clear the file table reference in the u-area. When a vhangup() occurred on a tty line which was being opened, it caused the following panic: Protection Fault. This problem has been resolved.	–
ifconfig	When /etc/ifconfig ln0 down failed, the network continued to run.	–
Local Area Terminal (LAT)	The ULTRIX system could send the terminal server a short start slot, causing node names to be truncated (for example, the command show sessions displayed node ABCDEF as AB). Also, node names less than four characters were rejected as illegal.	–
Local Area Terminal (LAT)	The following panics have been resolved: "unaligned access" – caused by receipt of a large status message "lock retry limit exceeded" - caused if many print jobs were active at once. "lock not locked" and "bad mcltype" – caused if a bad slot were received	–
Local Area Terminal (LAT)	Because vhangup was not being managed properly, it was possible to start a LAT session and end up connected to a process from a previous session (rather than being connected to getty, as would be the usual case). This would be seen with processes set to ignore SIGHUP.	–
Network File System (NFS)	A deadlock condition in the kernel routine vop_link() caused the nfsd to hang. This problem has been resolved.	–

Table A-1: (continued)

Component	Problem Resolved	CLD/SPR
Network File System (NFS)	Changes were made in the ULTRIX/UWS Version 4.1 NFS client and server that significantly improve performance when randomly patterned writes that are less than the filesystem blocksize (usually 8K) are done on NFS mounted filesystems.	–
Network Time Protocol (NTP)	The problem described in ULTRIX Version 4.0 release note 2.12.8, Network Time Protocol (NTP): Missing /etc/ntp.conf File, has been resolved.	–
qe(4)	In ULTRIX/UWS Version 4.0, the delqa device driver caused the network to hang on a DECsystem 5400. This problem has been resolved.	–
rexecd	The rexecd daemon did not work for normal users, only for root.	–
ruserok(3)	The return value for success was changed from 1 to 0.	–
Scheduling Problem	The problem that occasionally resulted in improper scheduling on heavily loaded systems has been resolved. Process priority adjustment could leave the scheduling queue (run queue) in a state where runnable processes were not in priority order; this could result in improper operation of round-robin scheduling on systems with numerous runnable processes.	–
SCSI Tapes	The problem described in ULTRIX Version 4.0 release note 2.13.16, Archiving Problems with SCSI Tapes, has been resolved.	–
tar	The problem with the -r and the -s options has been resolved. The tar(1) command now properly appends and updates files to a tape archive under all conditions. Previously, there was a one in twenty chance that tar(1) would incorrectly append the new file past the logical end of the tape archive. This only occurred when writing to a tape device; updates and appends to archives stored in a file were always correctly handled.	CLD 03445
tar	The tar(1) command now correctly handles file names of 100 characters. The previous version of tar(1) would compute the file name length based on the absolute pathname. The fixed version computes the file name length based on the name specified by the command line which, in most cases, is a relative pathname.	SPR 00823
ufs_rwgp()	The kernel routine ufs_rwgp() was protected from a negative file offset which caused the following panic: "Protection Fault".	–

Table A-1: (continued)

Component	Problem Resolved	CLD/SPR
ULTRIX/SQL	In ULTRIX/UWS Version 4.0, ULTRIX/SQL provided the file /usr/kits/sql/sqlstartup.rclocal for you to include in your /etc/rc.local file to enable the automatic startup of SQL at system boot time. However, in that version, this file failed to restart the DBMS servers properly after the system was taken down to single-user mode and then brought back up to multi-user mode without being rebooted. This problem has been corrected. If you are updating your system from ULTRIX/UWS Version 4.0 to ULTRIX/UWS Version 4.1 you might want to include the new version of the ULTRIX/SQL startup file in /etc/rc.local.	–
virtual memory subsystem	The following panic, caused by a process which does asynchronous (n-buffered) I/O to a raw device and also calls the vfork() system service, has been resolved: panic: "MUNLOCK: dup page unlock".	CLD OGO-04179
virtual memory subsystem	The following panic, caused by a process doing I/O between a raw device and shared memory, has been resolved: "panic: MUNLOCK: dup page unlock"	CLD IPO_03325
virtual memory subsystem	The following panic, caused by processes which read from a raw device to shared memory, has been resolved: "panic: vtopte SMEM" for processes	CLD IPO_03676
ypbind	The Initial Bind option (-X) has been added to ypbind. This option causes ypbind to attempt to bind to a YP server before backgrounding itself. If ypbind fails to bind to a YP server after three attempts it will exit. With the -X option a system that is not dependent on YP will not hang because there were no YP servers available at boot time.	SPR 671

A.2 ULTRIX Worksystem Software Problems Resolved Since Last Release

This section discusses problems in ULTRIX Worksystem Software that have been resolved since the last release. Table A-2 contains lists the topic or the name of the component that has been resolved, a definition of the problem that has been resolved, and, when applicable, a reference to a Software Performance Report (SPR) or Customer Level Distribution (CLD).

Table A-2: UWS Problems Resolved Since the Last Release

Component	Problem Resolved	CLD/SPR
DPS	A problem in output buffering routines has been fixed. Certain sequences of Display PostScript requests caused the client program to crash with a bad pointer reference. The content of the requests did not matter, but the length did. Adding `XFlush()` calls changed the behavior.	–
DPS	A problem in context memory reclamation has been fixed. The `DPSDestroyContext` routine would never destroy the context; the context would be rendered unusable, but its resources were not reclaimed.	–
DPS	A problem in color rendering has been fixed. The `setXrgbactual` operator would sometimes exhibit roundoff errors such that the displayed color was slightly different from the requested color.	–
DPS	An error in the spacing of text that caused text displayed using prebuilt fonts to be spaced too closely together has been fixed.	–
DPS	Text ligatures, characters of two or more letters joined together, are now properly displayed using prebuilts. Previously, they were rendered from outlines.	–
DPS	Several problems relating to Display PostScript and backing store have been fixed; notably, output into an obscured window with `backingStore` set to `Always` would not be saved.	–
DPS	The `colorimage` operator was not working correctly. Images with one or two bits per sample on an eight-plane display were sometimes rendered incorrectly.	–
DPS	A problem in the client library that caused incorrect updating of user names with shared or chained contexts has been fixed.	–
DPS	Prebuilt fonts were never being used on 100 dpi monitors. This has been fixed.	–
DPS	A problem involving `uappend` has been fixed.	–

Table A-2: (continued)

Component	Problem Resolved	CLD/SPR
DPS	A problem involving the strokepath operator has been fixed.	–
driver	Graphics and console drivers now cooperate, enabling the use of input devices such as graphics tablets.	–
dxcalendar	The dxcalendar application no longer prints an	–
dxcalendar	Double clicking the dxcalendar icon no longer crashes the application.	–
dxcalendar	Specifying a calendar database file other than .dxcalendar in the environment variable DXCALENDAR_FILE The dxcalendar application works across a VMS/ULTRIX connection.	SPR ICA-26778
dxcalendar	The dxcalendar application now correctly aligns icons and menu items when using 100dpi fonts.	SPR ICA-26300
dxcardfiler	The dxcardfiler application no longer exits prematurely when run under MOTIF.	SPR ICA-26708
dxdb	The dxdb application no longer goes into an infinite loop when executed from a bourne shell.	SPR ICA-25410
dxdiff	A problem involving random highlighting of text within xdiff is fixed.	–
dxdiff	The dxdiff application no longer suffers from segmentation faults.	–
dxpsview	The dxpsview application no longer displays the wrong copyright date.	–
dxsession	The dxsession application no longer crashes when dumping the screen to a color printer utilizing the sixel data format.	–
dxsession	A bug causing ‘´ characters to be stripped out of application startup commands has been fixed.	–
dxsession	The Session Manager properly executes a Print Screen command when the output format is Sixel.	SPR ICA-24882
dxsession	YP workarounds have been removed from the Session Manager.	–
dxsession	X11 Release 4 servers now allow users to login through Xprompter.	–
dxsession	Bits in the Session Manager window are no longer corrupted by overlapping pull-down menus.	–
dxsession	Specifying the client UID in the VMS Session Manager’s Customize Security menu now allows interoperability between a VMS host and an ULTRIX client.	–
dxterm	DECterm now accepts ctrl-3 and ctrl-] as input.	–

Table A-2: (continued)

Component	Problem Resolved	CLD/SPR
dxterm	DECterm no longer hangs during scrolling.	–
dxterm	DECterm now correctly displays Host Status information in the Host Status Display line at the bottom of the terminal window.	–
dxterm	DECterm now sends the correct leading character (a DCS) when generating a VTX color table report.	–
dxue	A bug that crashed the User Executive in the Double Click Actions menu under the Customize menu has been fixed.	–
install	The copyright symbol is now correctly displayed under the About menu selection in DECwindows applications' help menus.	–
install	ULTRIX/UWS Version 4.1 has been linked with the latest version of DECnet-ULTRIX `libdnet.a`.	–
server	The `Xcfb` server no longer suffers from segmentation faults.	–
Xlib	A problem involving Xlib's reporting of protocol errors is fixed.	–
Xlib	Allocation errors (*alloc errors) are now handled correctly by Xlib.	–
xdvi	The `xdvi` application no longer crashes the server.	–

Changes and New Features in Version 4.1 B

This appendix discusses changes and new features in the ULTRIX operating system and ULTRIX Worksystem Software that apply to ULTRIX/UWS Version 4.1.

B.1 ULTRIX Changes and New Features

This section discusses the following changes and new features in the ULTRIX operating system for ULTRIX/UWS Version 4.1:

- Conformance to standards and specifications
- Compatibility with earlier versions of the operating system
- Support for new processors
- Support for new hardware devices
- New software components
- New and changed documentation components
- New and changed software services for customers
- Software features no longer supported by the operating system
- Hardware no longer supported by the operating system.

B.1.1 Conformance to Standards and Specifications

ULTRIX/UWS Version 4.1 conforms and complies to several industry standards, including the following:

- POSIX 1003.1-1988, Portable Operating System Interface for Computer Environments, September 1988.
- FIPS Publication 151-1, POSIX: PORTABLE OPERATING SYSTEM INTERFACE FOR COMPUTER ENVIRONMENTS.
- X/Open Portability Guide, Issue 3, (December 1988) BASE OS:
  - Volume 2, XSI System Interface and Headers
  - Volume 3, XSI Supplementary Definitions (excluding curses interface)
  - Volume 4, Programming Languages

Certification of compliancy is accomplished through the use of the X/Open "Branding" process as well as the use of an accredited National Computer Systems Laboratory certified testing laboratory for POSIX/FIPS.

B.1.2 Compatibility With Earlier Versions of the Operating System

ULTRIX/UWS Version 4.1 is compatible in most ways with earlier versions of the ULTRIX operating system. For more information on system compatibility, see note C.1.2, Porting Version 3.1 Applications to ULTRIX/UWS Version 4.0.

B.1.3 New Processors

The following new processors are supported in ULTRIX/UWS Version 4.1:

DECsystem 5100	The DECsystem 5100 is a low-end desktop server, the follow-on to the DECsystem 3100.
DECsystem 5500	The DECsystem 5500 is a follow on to the DECsystem 5400.

B.1.4 New Hardware Devices

The following new devices are supported in ULTRIX/UWS Version 4.1:

RX23	The RX23 device is a 3.5 inch SCSI floppy disk that can hold up to 1.44 Mbytets of data.
RX33	The RX33 device is a 5.25 inch SCSI floppy disk that can hold up to 800 Kbytes of data.
RZ23L	The RZ23L device is a half–height 3.5 inch disk that can hold up to 120 Mbytes of data.
TLZ04 (RDAT)	The TLZ04 (RDAT) device is a 4mm tape that conforms to the DDS format. This device can hold up to 1.2 Gbytes of data.
TZK08 (EXCABITE)	The TZK08 (EXCABITE) device is a 8mm tape that can hold up to 2.3 Gbytes of data.
TZK10 (QIC)	The TZK10 (QIC) device is a standard 1/4" tape cartridge that conforms to the industry QIC format. The ULTRIX operating system will read and write QIC-320, QIC-150, and QIC-120. The ULTRIX operating system will only read QIC-24.
TZ30	The TZ30 device is a half–height SCSI TK50.

B.1.5 Software Component Features

The following software component feature is new in ULTRIX/UWS Version 4.1.

B.1.5.1 ULTRIX System Configuration and Management Program (SCAMP) – The ULTRIX System Configuration and Management Program (SCAMP) is a menu-driven program that helps you set up and manage your system. SCAMP allows you to manage user accounts and software subsets, and to perform basic system management tasks such as setting the system date and time, and rebooting the system.

Note, however, that the terminal and modem line setup menus will only create a terminal or modem line on `tty00`. The printer line setup menu will only create a printer line on `tty01`.

SCAMP can also be used for ongoing system management.

If you have installed a new system with ULTRIX/UWS Version 4.1 and your system configuration needs are basic then you may want to use SCAMP. Minimal computer experience and little UNIX or ULTRIX experience is required to use SCAMP.

The *Guide to SCAMP* explains how to set up your workstation and illustrates how to select the appropriate configuration information. It also provides references to guides in the ULTRIX documentation set where you can find more information on specific topics.

B.1.6 Documentation Component Features

The following document is new in ULTRIX/UWS Version 4.1:

Guide to SCAMP — Explains how to use the ULTRIX System Configuration and Management Program (SCAMP), a menu-driven program that helps you set up and manage your system.

The following documents have been changed in ULTRIX/UWS Version 4.1:

Guide to Configuration File Maintenance
Contains support for the new processors supported in ULTRIX/UWS Version 4.1.

ULTRIX/UWS Release Notes
Combines and reorganizes the ULTRIX and UWS release notes into one document.

ULTRIX/SQL Installation Guide
Adds support for the installation of the latest version of ULTRIX/SQL.

For a list of new and changed reference pages in ULTRIX/UWS Version 4.1, see the section on Reference Pages in Chapter 6.

B.1.7 Customer Services Component Features

System integrated and individual software services are available for ULTRIX/UWS Version 4.1. Individual services include telephone support, installation services, and media and documentation update services. For more information on these and new services for ULTRIX, contact your local sales office.

In addition, the following three services have been added since ULTRIX/UWS Version 4.0:

- System Management Service (SMS)

  SMS provides single point of access to a customer support center and a proactive problem resolution process.

- Software Update Installation Service (SUIS)

  SUIS provides a software update installation at a customer site by Digital specialists, who also explain the changes in the new release of the software.

- Source Code Update Service (SCUS)

  SCUS provides automatic updates of source code to customers with each scheduled version release of the operating system.

B.1.8 Software Features No Longer Supported

All software components supported in ULTRIX/UWS Version 4.0 remain supported in ULTRIX/UWS Version 4.1.

B.1.9 Hardware No Longer Supported

All hardware components supported in ULTRIX/UWS Version 4.0 remain supported in ULTRIX/UWS Version 4.1.

B.2 ULTRIX Worksystem Software Changes and New Features

This section discusses the following changes and new features in the ULTRIX Worksystem Software for ULTRIX/UWS Version 4.1:

- Xlib compatibility with MIT X11 Release 4
- Shared-Memory Transport (SMT) support for DECstation/DECsystem 5000 Model 200 series processors
- Three-dimensional graphics support for DECstation/DECsystem 5000 Model 200 series processors
- Changes to Applications

B.2.1 Release X11R4 Xlib support

The new Xlib library fixes several bugs in the resource manager and ICCCM convenience routines. Xlib is now fully compatible with MIT X11 Release 4. Applications using Xlib should be relinked with the MIT X11 Release 4 of Xlib contained in this release.

B.2.1.1 Undocumented Xlib Functions – Only documented Xlib functions are supported by the library. The use of undocumented functions produces undefined results and undermines application portability.

B.2.1.2 Xlib Size – The Xlib shipped with ULTRIX/UWS Version 4.1 (`/usr/lib/libX11.a`) requires more disk space than previously shipped versions. The increased `libX11.a` file size does not affect the final size of programs built using Xlib.

B.2.1.3 Xlib Function Declarations – The Xlib header files `Xlib.h` and `Xutil.h` now declare most Xlib external functions. Include these files in Xlib application code to ensure that Xlib functions are declared correctly. Applications which declare these functions differently will not link properly with the Xlib library.

B.2.2 Shared-Memory Transport (SMT) Support for DECstation/DECsystem 5000 Model 200

The DECstation/DECsystem 5000 PX, PXG, and PXG-turbo graphics options running ULTRIX/UWS Version 4.1 or later support Shared-Memory Transport (SMT) between X11 clients and the server. This section describes the components of the transport, system configuration parameters affecting the transport, and guidelines for

using the transport (SMT improves the performance of some applications more than others).

With SMT, X requests are built into a data segment shared between the client and server. This allows the server to start processing the request the moment it is complete, and prevents the request from being copied twice (client to kernel to server), as occurs with UNIX-domain sockets. If the memory bandwith is M megabytes/seconds, and there are N words in the packet, the time saved (excluding cache effects) is:

$$\text{Seconds saved per request} = \frac{(2 \times N)}{M} - \textit{Overhead}$$

In this equation, *Overhead* is fixed SMT per-request overhead. When N is large and the requests take relatively little time to execute compared to copy time, performance improves significantly. Even when a particular application experiences little or no performance improvement, idle time and bus utilization are shorter than with UNIX-domain sockets. Thus overall system throughput is improved.

B.2.2.1 Using the SMT Extension – For an application to use SMT, the following requirements must be met:

- The application must be linked against an Xlib which supports the SMT extension. All versions of Xlib shipped with ULTRIX/UWS Version 4.1 or later support SMT.
- The application must run on the same machine as the server.
- The application must be invoked with a command line option setting `-display` to "`local:0`".
- The server on the local machine must support the SMT extension.

B.2.2.2 Guidelines for Using SMT – If you are considering using SMT with an existing application, the most expedient thing to do is to try it with SMT. When assessing performance, the `time` command should be used to measure system resource use. In many cases, system overhead will decrease even if performance remains constant, leading to a higher overall system throughput. Because SMT trades off the time to transmit requests to the server against higher per-request overhead, some applications may realize no gain (or even a small decrease) in performance. Therefore, it is important to measure the performance of critical applications both with and without SMT. If you are developing a new application, the following guidelines will help you maximize the benefit gained by using SMT:

- SMT is intended for performance-critical applications. It is inadvisable to make SMT the default transport.
- In the current implementation, only requests use SMT. The performance of applications which receive large amounts of data back from the server (for example, `GetImage`) is not significantly affected by SMT.
- Applications should pack as many primitives into an X request as possible, and call `XFlush` only when necessary. This is good advice to follow when writing any X application, but it applies particularly to SMT.

- UNIX-domain sockets only buffer 4 kilobytes of data before flushing to the server; SMT buffers up to 256 kilobytes of data. Applications which assume server flushes occur frequently should add additional `XFlush` calls.
- Requests most likely to benefit from SMT are `PutImage`, `PolyPoint`, `PolySegment` (solid zero-width), and `PolyFillRectangle` (where the rectangles are small). Requests least likely to benefit from SMT are `GetImage`, wide primitives, and window management requests (circulate, resize, and so on).

B.2.2.3 SMT Usage Limits – An X11 server can provide a limited number of SMT connections to clients. The actual number allowed is defined in the kernel configuration file. `SMSEG` defines the maximum number of SMT links available. Two of these links are used by the server, leaving a total of (`SMSEG` –2) available for use by applications. The default value for `SMSEG` is 6, so the default number of SMT connections that an X11 server can support is 4.

The maximum size of a shared memory segment is defined in the kernel configuration file by `SMMAX`. The default value for `SMMAX` is 1024, which is adequate for SMT.

B.2.2.4 SMT Error Messages – Improper use of SMT connections may result in the display of one or more of the following diagnostic error messages:

`X Toolkit Error: Can't Open display`
This means that the application is linked against an Xlib which does not support the SMT extension. Relink the application against an Xlib with SMT support.

`SMT-WARNING: extension not supported,...`
This means that the application has SMT support, but that the server does not. You should not receive this message on systems equipped with the PX, PXG, or PXG-turbo graphics options.

`SMT-WARNING: local shmem attach failed,...`
This means that the client was unable to create a shared memory segment. The most likely cause of this is either that the process is not able to attach any more segments (system config parameter `SMSEG` too low), or that a sufficiently large segment could not be created (system config parameter `SMMAX` too small).

`SMT-WARNING: server refused attach,...`
This means that the client was able to successfully create a shared memory segment, but that the server was unable to attach the segment. The most likely explanation is that the server cannot accept any new SMT clients because it has already attached the maximum number of shared memory segments. Before raising the smseg system config parameter, verify that inappropriate clients are not monopolizing the limited number of SMT connections available to the server.

`SMT-ERROR: <extension> incompatible with shared-memory, ...`
This means the application is linked with a server extension which does not support SMT. This will cause the application to terminate. The Display Postscript, PHIGS, and multibuffering extensions shipped with the DECstation/DECsystem 5000 support SMT; non-DIGITAL extension libraries probably do not.

B.2.3 Three-Dimensional Graphics Support for DECstation/DECsystem 5000 Model 200

Through DECphigs, users can access three-dimensional graphics features of the DECstation/DECsystem 5000 Model 200PX, 200PX, and 200PXG-turbo. Refer to your DECphigs manual for more information on how to take advantage of these features. Platform specific release notes for DECphigs are in the processor-specific chapter of these release notes.

B.2.4 Changes to Applications

This section describes changes to ULTRIX Worksystem Software applications.

B.2.4.1 DECterm Changes – The following notes pertain to the DECterm terminal emulator, `dxterm`.

B.2.4.1.1 DECterm Grey Levels on Monochrome Systems – The eight default grey levels used by DECterm on monochrome systems have been lightened. The new levels are more visible and easier to distinguish from one another. The former levels were suited to the display of a VT340, rather than to a windowing display terminal.

B.2.4.1.2 DECterm ReGIS Locator Reporting – If the ReGIS locator position is outside the range of addressable locations, the locator report returns a null position [ ].

B.2.4.1.3 DECterm Conformance Level Checking – DECterm now allows terminal state reporting when operating at conformance level 2 or conformance level 3. Previously, DECterm allowed terminal state reporting when operating at conformance level 3 only. Conformance level 1 still inhibits terminal state reporting from DECterm.

B.2.4.1.4 DECterm Answerback – A bug that prevented answerback from working has been fixed. In response to the '^E' character, DECterm now responds with the value (if any) of the answerbackMessage resource.

B.2.4.1.5 DECterm VT52 Mode Cursor Addressing – Cursor addressing while in VT52 emulation mode is no longer limited to a fixed row or column length.

B.2.4.1.6 DECterm Conformance Level Report Escape Sequence – The escape sequence used by DECterm (operating at conformance level 1 only) when generating a conformance level report is now as follows:

```
<DCS>1$r"p<ST>
```

The former escape sequence included a suffix that was invalid. Custom application software that is coded to expect the invalid suffix must be changed to reflect this correction.

B.2.4.1.7 DECterm Color Table Report Prefix – DECterm now correctly prefixes color table reports with a `DCS` (ASCII code 144) instead of a `CSI` (ASCII code 155). Custom application software coded to expect a `CSI` must be changed to reflect this correction.

B.2.4.1.8 DECterm Memory Limitations – DECterm now ignores requests to resize its window and off-screen text buffer when additional memory is unavailable.

B.2.4.2 Visual Differences Program - dxdiff – The following `dxdiff` features have been added since ULTRIX/UWS Version 4.0:

- File selection dialogs are now unmanaged when files are selected.
- `dxdiff` no longer intermittently dumps core when successive Do Difference commands are issued.
- When two identical files are compared after two different files, `dxdiff` no longer displays random highlights.
- Scrolling with MB2 (toTop) and MB3 (toBottom) is now supported in `dxdiff` scrollbars.

Changes and New Features in Version 4.0 C

This appendix discusses changes and new features in the ULTRIX operating system and ULTRIX Worksystem Software that apply to ULTRIX/UWS Version 4.0.

C.1 ULTRIX Changes and New Features

This section discusses the following changes and new features in the ULTRIX operating system for ULTRIX/UWS Version 4.0:

- Conformance to standards and specifications
- Compatibility with earlier versions of the operating system
- New processor support
- New hardware device support
- New and changed software components
- New and changed documentation components
- New and changed software services for customers
- Software features no longer supported by the operating system
- Hardware no longer supported by the operating system.

C.1.1 Conformance To Standards and Specifications

This section defines how the release conforms to industry standards and external specifications.

C.1.1.1 Industry Standards Conformance – Although ULTRIX/UWS Version 4.0 will be fully conformant to all the Standards listed below, official compliancy verification procedures will not be available before ULTRIX/UWS Version 4.0 ships.

Certification of compliancy will be accomplished through the use of the X/Open "Branding" process as well as the use of an accredited National Computer Systems Laboratory certified testing laboratory for POSIX/FIPS.

The standards are:

- POSIX 1003.1-1988, Portable Operating System Interface for Computer Environments, September 1988.
- FIPS Publication 151-1, POSIX: PORTABLE OPERATING SYSTEM INTERFACE FOR COMPUTER ENVIRONMENTS.
- X/Open Portability Guide, Issue 3, (December 1988):
  - Volume 2, XSI System Interface and Headers

– Volume 3, XSI Supplementary Definitions (excluding curses interface)

– Volume 4, Programming Languages

C.1.1.2 Changes to Header Files – A number of header files have been changed to conform to one or more of the following standards:

- American National Standard for Information Systems– Programming Language C (ANSI C)
- IEEE Standard Portable Operating System Interface for Computer Environments, IEEE Std 1003.1-1988 (POSIX)
- X/Open Portability Guide: XSI System Interface and Headers, Issue 3 (XPG 3)

C.1.1.2.1 Typical Program Changes – Programs may need changes because of this. Typical changes include:

- Removing redundant, outdated declarations of functions (those in `stdio.h` have been most common; for example, `fread`).
- Renaming of conflicting internal functions (for example, `remove`, which is now defined in `stdio.h`).
- Moving multiple inclusions of a header file within a function out to the file level; this may be needed because many header files are now protected against double inclusion, so a second inclusion of the same header file will not give a second definition of a structure.
- Altering data types or casts to conform to the new definitions (most notably, the change of many char * return values to void *).
- Altering some undocumented uses of macros; for instance, the fileno macro could formerly be used on the left side of an assignment statement, though this was neither documented nor supported; it can no longer be misused in this way.

In general, aside from the semantic changes above, these problems will show up as compile-time errors or warnings, thus making them easy to spot.

C.1.1.2.2 Specific Header File Changes – The following list defines changes to the header files:

<assert.h>

Now calls the __assert function (unless NDEBUG is defined) in all cases; prints the text of the failed assertion.

<ctype.h> (and <sys/ctype.h>)

Added definitions for tolower(), toupper().

<errno.h> (and <sys/errno.h>)

Added definition for ENOSYS.

<fcntl.h> (and <sys/file.h>)

Added definitions for creat(), fcntl(), open().

<float.h>

Definitions now present for both VAX and RISC architectures. Some values have been adjusted.

<ftw.h>

Added ftw() declaration.

<grp.h>

Double-inclusion protection.

<limits.h> (and <sys/limits.h>)

Added definitions for NZERO, MB_LEN_MAX, NL_NMAX. Corrected several floating point values.

<locale.h>

Added definitions for NULL, struct lconv, localeconv().

<math.h>

Added definitions for strtod(), lgamma(), isnan(), atol(); corrected VAX architecture definition of HUGE.

<memory.h>

Added memmove(); changed return values from char * to void *. Note that, for new applications, the <string.h> header should be used instead.

<nl_types.h>

Added definitions of NL_SETD, catclose(), catgets(), catopen().

<pwd.h>

Double-inclusion protection.

<search.h>

Added definitions for hsearch(), hcreate(), hdestroy(), tsearch(), tfind(), tdelete(), twalk(), lsearch(), lfind().

<setjmp.h>

Double-inclusion protection; added definitions for setjmp(), longjmp() (RISC only; they were already present for the VAX architecture). Added definitions for sigsetjmp(),siglongjmp().

<signal.h> (and <sys/signal.h>)

Added definitions for BRK_STACKOVERFLOW, sig_atomic_t, raise(), kill(), sigaction(), sigaddset(), sigdelset(), sigemptyset(), sigfillset(), sigismember(), sigpending(), sigprocmask(), sigsuspend().

<stdarg.h>

Added definitions for VAX architecture.

<stddef.h>

New file.

<stdio.h>

Double-inclusion protection; added definitions for size_t, FOPEN_MAX, FILENAME_MAX, TMP_MAX, fpos_t, SEEK_SET, SEEK_CUR, SEEK_END, ctermid(), cuserid(), fgets(), gets(), tempnam(), tmpnam(), fclose(), fflush(), fgetc(), fgetpos(), fprintf(), fputc(), fputs(), fscanf(), fseek(), fsetpos(), getw(), printf(), puts(), putw(), remove(), rename(), scanf(), setvbuf(), sscanf(), ungetc(), vfprintf(), vprintf(), vsprintf(), fread(), fwrite(), clearerr(), perror(). Note that the definitions of fread() and fwrite() are now size_t (an unsigned int) instead of the previous int.

<stdlib.h>

New file. Definitions of calloc(), free(), malloc(), realloc(), bsearch(), and qsort() differ from their previous return values.

<string.h> (and <strings.h>)

Added definitions for size_t, NULL, strerror(), strstr(), memcmp(), strcoll(), strxfrm(), memccpy(), memchr(), memcpy(), memmove(), memset(); note that the definitions of strcspn(), strlen(), strspn(), and strxfrm() are now size_t (an unsigned int) instead of the previous int, and the mem functions are now void * instead of char *.

<time.h> (and <sys/time.h>)

Added definitions for NULL, time_t, clock_t, CLOCKS_PER_SEC, CLK_TCK, strftime(), clock(), time(), mktime(), difftime(). Note that clock(), time(), and mktime() have new return types defined. Note also that the definition of size_t has been changed from int to unsigned.

<ulimit.h>

New file.

<unistd.h>

Added definitions for _SC_XOPEN_VERSION, _SC_PASS_MAX, _XOPEN_VERSION, STDIN_FILENO, STDOUT_FILENO, STDERR_FILENO, access(), chdir(), chown(), close(), dup(), dup2(), execl(), execle(), execlp(), execv(), execve(), execvp(), getgroups(), isatty(), link(), pause(), pipe(), read(), rename(), rmdir(), setgid(), setpgid(), setuid(), tcsetpgrp(), unlink(), write(), fpathconf(), pathconf(), sysconf(), ctermid(), cuserid(), getcwd(), getlogin(), ttyname(), _exit(), alarm(), sleep(), getegid(), getgid(), lseek(), fork(), getpgrp(), getpid(), getppid(), setsid(), tcgetpgrp(), geteuid(), getuid(). The functions from alarm() on have different definitions than they did in some earlier releases; except for alarm() and sleep(), though, these are merely typedefs for the existing practice.

Added inclusion of <sys/types.h>.

<utime.h>

Added definition for utime().

<sys/stat.h>

Double-inclusion protection; added definitions for umask(), chmod(), fstat(), mkdir(), mkfifo(), stat(). Note that umask is now defined to return mode_t.

<sys/types.h>

Changed definition of size_t from int to unsigned.

<sys/utsname.h>

Double-inclusion protections; added definition for uname().

<sys/wait.h>

Double-inclusion protection; added definitions for wait(), waitpid(); added inclusion of <sys/types.h>.

C.1.2 Porting Version 3.1 Applications to ULTRIX/UWS Version 4.0

ULTRIX/UWS Version 4.0 is compatible in most ways with earlier versions of ULTRIX. This compatibility allows you to easily port your programs from Version 3.1 to ULTRIX/UWS Version 4.0. (In this section, references to Version 3.1 include Version 3.1B, Version 3.1C, and Version 3.1D.) Most of your applications will run unchanged under an appropriate configuration of ULTRIX/UWS Version 4.0. This section explains how to port your applications by describing the levels of portability available in ULTRIX/UWS Version 4.0 and the program features that affect portability.

C.1.2.1 Levels of Portability – In describing levels of portability, this section assumes that you are porting an application to the same hardware architecture; that is, that you are porting from Version 3.1 on a VAX machine to ULTRIX/UWS Version 4.0 on a VAX machine, or from one RISC machine to another. For information on porting from a VAX machine to a RISC machine, see Appendix D.

The section also assumes that you are porting to the same programming environment you used on Version 3.1. For example, if you used the BSD environment (the default) on Version 3.1, you should port to the BSD environment (the default) on ULTRIX/UWS Version 4.0. Likewise, if you used the POSIX or System V environment on Version 3.1, you should use that environment on ULTRIX/UWS Version 4.0.

Depending on the ULTRIX ULTRIX/UWS Version 4.0 subsets you have installed and the operating system features your program uses, you can achieve one of the following three portability levels:

- Binary portability

  In most cases, a binary (a.out) executable you created on a Version 3.1 system will run on ULTRIX/UWS Version 4.0. If you install the security or the Distributed System Services (DSS) subsets on your system, you will be unable to run some binary executables on ULTRIX/UWS Version 4.0, even though these executables ran correctly on Version 3.1. Refer to Section C.1.2.2.6 for more information.

- Object file portability

  You can move object files generated on Version 3.1 systems to an ULTRIX/UWS Version 4.0 system and link them with the appropriate ULTRIX/UWS Version 4.0 libraries to achieve another level of portability. In this case, many new features in the libraries will be available for you to use. For information on the new features, see the *ULTRIX Reference Pages, Section 3: Library Routines*. You must relink your programs if you have installed and enabled the new C2 security features or DSS subsets.

- Source code portability

  In some cases, you might have to modify or recompile your source code to port it to a ULTRIX/UWS Version 4.0 system. You must recompile your source code and possibly modify it if you want to use new features or if your application accesses modified data structures.

C.1.2.2 Program Features that Affect Portability – This section describes ULTRIX programming features that affect your ability to port your program to ULTRIX/UWS Version 4.0.

C.1.2.2.1 Direct Access of Kernel Data Structures – If your program accesses kernel data structures directly, you must modify your source code. Many ULTRIX internal structures have changed for ULTRIX/UWS Version 4.0. Fields in the structures have been added, modified, and removed.

Check the header files included in your program to determine how they have changed. Then, modify your source code and recompile your program.

C.1.2.2.2 Header File Changes – If you use certain header files, you must recompile your program to port it to ULTRIX/UWS Version 4.0. Check the header files included in your program. If a header file has changed, recompile your program.

C.1.2.2.3 File Protection Changes – The ULTRIX operating system contains two new groups that protect access to device special files. The `tty` group (gid 5) controls access to terminal devices in the `/dev` directory. The `kmem` group (gid 6) controls access to memory devices, such as the `/dev/mem` and `/dev/kmem` files.

Previously, `tty??` device files were given world write access by default. For ULTRIX/UWS Version 4.0, these files are given group write access and have a group owner of `tty`. The programs that write to terminals, such as `wall` and `write`, have been modified. These programs now set their gid to the tty group on execution.

If your program reads or writes the memory files in the `/dev` directory, you must modify it to set the group to kmem on execution. The following shows an example of changing the group and mode of a program named `user_program:`

```
% chgrp kmem user_program
% chmod g+s user_program
```

Only trusted programs should have access to the kmem and tty groups. For more information on writing secure programs, see the *Guide to Languages and Programming* .

C.1.2.2.4 Optional New Password File Format – In ULTRIX/UWS Version 4.0, system administrators can create a 4.3BSD-style hashed password database. If the hashed password database exists, the system will use it instead of `/etc/passwd` for all calls to `getpwuid` and `getpwnam.` For more information about the new hashed database, see `mkpasswd`(8).

If a system administrator edits the `/etc/passwd` file without using the `vipw` command, the `/etc/passwd` file might be different from the hashed password database. Because the hashed password database and the `/etc/passwd` file can be different, a Version 3.1 program that reads only the `/etc/passwd` file might produce different results than a ULTRIX/UWS Version 4.0 program that reads the hashed password database. System administrators can use the `mkpasswd` file command to resolve inconsistencies between the hashed password database and the `/etc/passwd` file.

C.1.2.2.5 Optional Security Subset – If your system takes advantage of the new C2 security features, you might need to modify your source code to move your Version 3.1 program to ULTRIX/UWS Version 4.0. You might need to modify your source code if your program reads the password file to get user password information.

When setting up the secure system environment, a system administrator chooses whether the system uses the BSD login features or the new secure login features. If your ULTRIX/UWS Version 4.0 system uses the BSD login features, you need not modify your program. The ULTRIX/UWS Version 4.0 BSD password features are compatible with the Version 3.1 password features.

If the system administrator chooses the new secure login features, you must modify your program if it needs user password information. For security reasons, passwords in the secure login environment are stored separately from other user information. For information about authenticating users in a secure environment, see the *Guide to Languages and Programming* .

C.1.2.2.6 Distributed Environment Changes – In ULTRIX/UWS Version 4.0 you can use BIND/Hesiod as an alternative to Yellow Pages (YP). The binaries of these components have been built with new libraries that provide transparent access to distributed configuration information. A new file, `/etc/svc.conf`, replaces the `/etc/svcorder` file. The `/etc/svc.conf` file allows you to configure different search paths for different types of system configuration data. This file contains one or more lines with the following format:

```
database=service,service
```

For information on the values you can use for *database* and *service*, see the *Introduction to Networking and Distributed System Services* .

To make your ULTRIX/UWS Version 4.0 system compatible with Version 3.1, you must provide the same information access configuration in ULTRIX/UWS Version 4.0 as you had in Version 3.1. You can use the `svc.conf` file to configure a number of binary compatible ULTRIX/UWS Version 4.0 systems.

The following three examples show entries in the `/etc/svc.conf` file that permit maximum binary compatibility:

- This example shows the default `/etc/svc.conf` file that causes all information access to be done on your local system:

```
# /etc/svc.conf installed settings

aliases=local
auth=local          # Note: the auth entry is not used in BSD
                    #     and is never used with yp
group=local
hosts=local
netgroup=local      # Note: the netgroup entry can only be used
                    #     with local or yp
networks=local
passwd=local
protocols=local
rpc=local
services=local
.
.
.
```

- This example shows how to set up your system with all information access either on your local system or through YP:

```
# modified /etc/svc.conf settings

aliases=local,yp
auth=local              # Note: the auth entry is not used in BSD
                        #     and is never used with YP
group=local,yp
hosts=local,yp
netgroup=yp             # Note: the netgroup entry can only be used
                        #     with local or YP
networks=local,yp
passwd=local,yp
protocols=local,yp
rpc=local,yp
services=local,yp
.
.
.
```

- This example demonstrates how to set up a ULTRIX/UWS Version 4.0 system so that local Version 3.1 settings are local on the ULTRIX/UWS Version 4.0 system; Version 3.1 settings that used YP use YP on the ULTRIX/UWS Version 4.0 system; and hosts served by BIND on the Version 3.1 system are served by BIND on the ULTRIX/UWS Version 4.0 system:

```
# modified /etc/svc.conf settings

aliases=local,yp

auth=local              # Note: the auth entry is not used in BSD
                        #     and is never used with yp
group=local,yp
hosts=local,bind
netgroup=yp             # Note: the netgroup entry can only be used
                        #     with local or yp
networks=local,yp
passwd=local,yp
protocols=local,yp
rpc=local,yp
services=local,yp
.
.
.
```

Several `libc` libraries have been modified to use the new `/etc/svc.conf` file. These libraries are: `libc.a`, `libcV.a`, `libcV_p.a`, `libcVg.a`, `libc_p.a`, and `libcg.a`. If your program will use BIND/Hesiod for any database except the hosts database on your ULTRIX/UWS Version 4.0 system, you must relink your program to these modules. You must also relink your program if it uses the `/etc/svc.conf` file and calls any of the following routines:

- `getprotobynumber`, `getprotobyname`, `getprotoent`, `setprotoent`, or `endprotoent`
- `getrpcbynumber`, `getrpcbyname`, `getrpcent`, `setrpcent` or `endrpcent`,
- `getnetbyaddr`, `getnetbyname`, `getnetent`, `setnetent`, or `endnetnet`
- `getgrgid`, `getgrnam`, `getgrent`, `setgrent`, or `endgrent`
- `getpwuid`, `getpwnam`, `getpwent`, `setpwent`, or `endpwent`
- `getservbyport`, `getservbyname`, `getservent`, `setservent`, or `endservent`
- `gethostbyaddr`, `gethostbyname`, `gethostent`, `sethostent`, or `endhostent`

C.1.2.2.7 Modified System Calls – The following system calls have been modified to support the POSIX standard:

- `msgrcv`
- `msgsnd`
- `msgctl`

If your program uses any of these calls, you must modify and recompile it. For more information about these calls, see the `msgrcv`(2) or `msgctl`(2) reference page.

C.1.2.2.8 New System Call Return Values – If your program expects a specific error to be returned from a system call or a library routine under a certain set of conditions, you might need to modify your program. A number of system calls return new or changed values. For example, the `sigvec` call return value has changed. This call no longer returns EINVAL when SIGCONT is ignored (given an action of SIG_IGN). Your program can now ignore the SIGCONT status.

For details on the values returned from each system call, see the *ULTRIX Reference Pages, Section 2: System Calls*.

C.1.2.2.9 Dependencies on Undocumented Features or Software Errors – You might need to modify your program if it depends on an undocumented feature or software error. Some undocumented features are no longer shipped in ULTRIX/UWS Version 4.0, and a number of software errors have been corrected.

For example, the `getpwent`(3) reference page states that the `setpwent` and `endpwent` calls must be used with `getpwent.` This statement is correct. A previous software error would allow you to use `getpwent` without the `setpwent` and `endpwent` calls. This error has been corrected, so if your program uses `getpwent` you must be sure it also calls `setpwent` and `endpwent.` This situation occurs with a number of other library routines, including `getprotoent`, `getrpcent`, `getnetent`, `getgrent`, `getservent`, and `gethostent`.

Read Appendix A carefully to determine what software corrections affect programs you are porting to ULTRIX/UWS Version 4.0.

C.1.2.3 POSIX and X/OPEN Programming Environments – ULTRIX/UWS Version 4.0 supplies a programming environment that allows you to write programs that conform to the POSIX and X/OPEN standards. The POSIX standard describes an interface to the ULTRIX operating system that eases porting programs from one system to another.

For information on programming in the POSIX and X/OPEN environments, refer to the Programming in a POSIX Environment chapter of the *Guide to Languages and Programming*. However, keep the following exceptions in mind:

- Whenever reference is made to the _POSIX_SOURCE symbol, it also applies to the new _XOPEN_SOURCE symbol.
- The libcP library provides functions for both POSIX and X/OPEN.
- Section 9.4, Compiling in the POSIX Environment, also supports X/OPEN, if the _XOPEN_SOURCE symbol is present within the application.

  Specifically, %PROG_ENV=POSIX applies to both environments and the -YPOSIX option on the `cc` command line sets the programming environment to both POSIX and X/OPEN.

C.1.2.3.1 POSIX Environment and Trusted Path Handling – If a trusted path is configured on your system, BREAK characters entered on any serial line result in the line being disconnected. Trusted path handling supersedes any special break character handling. This results dial-up lines.

In a POSIX environment, for example, the input mode IGNBRK will not function correctly. This is due to the fact that trusted path handling is performed in response to a BREAK character instead of ignoring the BREAK.

On systems that support modems, trusted path handling interferes with communications between the system and the modem, as well as with applications using the modem. For example, `uucico`(8) uses the BREAK character in its protocol for communicating between systems. When a BREAK character is received by a system supporting a trusted path, the line the `uucico` uses to process its communication is disconnected.

To work around these problems, do not configure trusted path on systems that support modems and/or run in a POSIX environment. To deconfigure a trusted path, follow these steps:

1. Remove the following lines from the system configuration file:

```
options  SYS_TPATH
pseudo-device  sys_tpath
```

2. Rebuild the kernel.
3. Reboot the system using the new kernel.

For information on rebuilding your kernel, see the *Guide to System Configuration File Maintenance*.

C.1.2.4 Correct Declaration for environ Global Variable – The correct declaration for the `environ` global variable has always been that stated in the `execl`(3) reference page:

```
extern char **environ;
```

In previous releases, a program that omitted the `extern` keyword in the declaration would probably still work. In ULTRIX/UWS Version 4.0, however, such a program is less likely to work because of structural changes in the C libraries.

To prevent problems, you should find and correct any declarations that incorrectly omit the `extern` keyword. The corrections are particularly important to apply to RISC-based code.

C.1.2.5 The /etc/group File Changes – New entries have been added to the `/etc/group` file. Some features of ULTRIX depend on these entries being present. Deleting or reusing these entries may also compromise the security of your system.

Deleting or renaming the "tty" group entry will cause the `write`(1) and the `wall`(1) commands to behave differently for nonprivileged users.

The "kmem" group's GID is reserved for applications requiring access to the system memory image.

The "authread" group is used to grant read access to the Auth Database when enhanced security features are enabled and is required for correct operation of these features.

C.1.2.6 Changes to tty Special File Defaults – The default group ownership and modes of tty special files set by `getty`(8) and `login`(1) have been changed to prohibit write access to the terminal by other than the owner and the group "tty." At the same time, the `write`(1) and `wall`(1) commands have been distributed as "setgid tty" programs. This change prevents anonymous writing to a user's terminal. The `mesg`(1) command will grant or deny group write access to the terminal but will not allow world access to it.

C.1.3 New Processors

This section lists new processors supported by ULTRIX/UWS Version 4.0. Note the name change of the VAX 62xx and VAX 63xx series of processors.

VAXstation 3540

DECstation and DECsystem 5000, Model 200

DECsystem 5400

DECsystem 5810, 5820, 5830, 5840

VAX 6000-220, 6000-230, 6000-240 (formerly the 6220, 6230, and 6240)

VAX 6000-330, 6000-340, 6000-350, and 6000-360 (formerly the 6330, 6340, 6350, and 6360)

VAX 6000-420, 6000-430, 6000-440, 6000-450, 6000-460

VAX 8300, 8350

VAX 8820, 8830, 8840

C.1.4 New Devices

This section lists new devices supported by ULTRIX/UWS Version 4.0.

DF296 - modem
KDM70 - XMI to SI disk/tape controller
RA92 - 1.5 Gigabyte disk drive
RF31 - 300 Mbyte half-height 5.25" form factor DSSI disk
RX23 - 1.4 Mbyte 3.5" double-sided diskette
RZ24 - 200 Mbyte 3.5" SCSI disk
RZ56 - 660 Mbyte 5.25" SCSI disk
RZ57 - 1+ Gigabyte 5.25" SCSI disk
TA90 - 200 Mbyte IBM compatible tape drive
TA90E - 400+ Mbyte IBM compatible tape drive, with compaction
TLZ04 - 1.3 Gigabyte scan tape, half height
TSZ05 - SCSI 9-track tape drive

C.1.5 Software Component Features

This section highlights the major new features and changes to the software.

C.1.5.1 Distributed System Services (DSS)–YP, BIND/Hesiod, Kerberos, timed, NTP – ULTRIX/UWS Version 4.0 product provides great flexibility in configuring a distributed environment. This section describes the distributed system services and their respective features.

C.1.5.1.1 Name Services – There is now an alternative service to Yellow Pages (YP). The Berkeley Internet Name Domain (BIND) service has been enhanced with Hesiod name service capabilities. The new Hesiod functionality allows the BIND service to distribute `aliases`, `auth`, `group`, `networks`, `passwd`, `protocols`, `rpc`, and `services`, in addition to `hosts` information. The library routines in `libc.a` have been modified to allow transparent access to BIND/Hesiod, YP, and local `/etc` files.

If the BIND/Hesiod service is used to distribute the `passwd` database, then the BIND/Hesiod primary server must have the alias ''bindmaster'' on its hosts entry in the hosts database. (The `bindsetup` script creates the alias.) This allows the BIND/Hesiod primary server to run the Hesiod password update daemon, `/usr/etc/hesupd`. The `passwd`(1) command in ULTRIX/UWS Version 4.0 has been enhanced to communicate password changes to the bindmaster's `hesupd`, thus providing a transparent distributed password program. The password changes take at most 4 minutes to propagate throughout the distributed environment.

For additional information on BIND/Hesiod, see *Guide to the BIND/Hesiod Service*.

C.1.5.1.2 Configurable Security Modes – The ULTRIX/UWS Version 4.0 product can be configured in different security modes. The choice of a security mode other than BSD (the default) dictates the use of BIND/Hesiod to distribute the `auth` and `passwd` databases. YP can only be used to distribute the `passwd` database in BSD security mode.

You must first set up a distributed BSD BIND/Hesiod environment before attempting to make the transition to either of the other two configurable security levels: UPGRADE or ENHANCED. The BSD distributed environment verifies that the BIND/Hesiod primary and secondary servers are all reachable and functioning properly.

Converting from a YP distributed environment to a BIND/Hesiod distributed environment or running a dual YP-BIND/Hesiod environment is possible in ULTRIX/UWS Version 4.0. The source files for BIND/Hesiod can now be `/etc` style files. Consequently a distributed BSD YP source area can be shared with symbolic links with a distributed BSD BIND/Hesiod area. If you are running at either UPGRADE or ENHANCED security levels, then the `passwd` and `auth` databases must be only distributed by BIND/Hesiod. When running in the UPGRADE or ENHANCED modes, the BIND/Hesiod daemon, `named`, should be set up to use Kerberos to prevent name server spoofing and possible breakins. If you are in a heterogeneous vendor environment, YP is the only choice for password distribution.

When running the Kerberos authenticated BIND/Hesiod daemon, `named`, zone transfers from non-authenticated Internet BIND daemons do not work. The work-around for this situation is to designate at least one system as a non-authenticated BIND secondary server of the external Internet domain. The local Kerberos authenticated BIND/Hesiod servers should be set up in `/var/dss/namedb/named.boot` to reference the non-authenticated BIND server as a forwarder. This allows the authenticated BIND daemons to access the external Internet domain through the forwarder.

In an ENHANCED environment all machines are either Kerberos servers or clients, or they are standalone.

C.1.5.1.3 Kerberos – The Kerberos authentication service has been integrated into the ULTRIX/UWS Version 4.0 BIND/Hesiod daemon, `named`. If you want to run a distributed UPGRADE or ENHANCED environment, you must run Kerberos. Both Kerberos and BIND/Hesiod use backup servers to provide performance and reliability. At least one backup (BIND/Hesiod secondary/Kerberos slave) server should be used in a distributed environment. It is strongly suggested that each BIND/Hesiod server (primary or secondary) be a Kerberos (master or slave).

The `passwd`, `login`, `su`, `dxsession`, and `named` programs have been modified to use Kerberos to access distributed `auth` data required by the UPGRADE or ENHANCED security levels.

Workstations and other less utilized machines in the distributed (UPGRADE or ENHANCED) environment must run as BIND/Hesiod slaves.

For information on setting up Kerberos, see *Guide to Kerberos*.

C.1.5.1.4 Network Time Protocol (NTP) and timed – ULTRIX/UWS Version 4.0 provides the ability to keep the distributed environment time synchronized via both `timed` and NTP. The NTP daemon, `ntpd`, should be run on the most secure and best administered machines.

The set of well administered NTP systems provides a base of secure time which will not drift at the whim of the other machines in the environment. If possible the master `ntpd` should reference a WWV radio clock for the utmost in accuracy. The remainder of the distributed environment runs `timed` without any options. The resultant combined `time` service is less vulnerable to the drifting nature of `timed` alone and is easier to administer than NTP alone.

Guide to System and Network Setup includes the procedures for running `ntpd` and `timed` in a distributed environment.

C.1.5.2 Packet Filter Pseudo-device Driver – A new feature in ULTRIX/UWS Version 4.0 is the packet filter pseudo-device driver, a kernel-resident network packet demultiplexer. The code appears to applications as character special files.

The packet filter provides a raw interface to Ethernets and similar network data link layers. Packets received that are not used by the kernel (for example, to support the IP and DECnet protocol families) are available through this mechanism.

In addition to the pseudo-device driver, ULTRIX/UWS Version 4.0 includes two commands for use with the packet filter. The `pfstat`(8) command prints packet filter status information. The `pfconfig`(8c) command allows system managers to configure the packet filter to enable non-privileged users to set an interface into or out of promiscuous mode, and to configure the packet filter input queue length.

Information on the packet filter is provided in the `packetfilter`(4) reference page, in the reference page for the two commands, and in the *Network Programming Manual*, all of which are provided with ULTRIX/UWS Version 4.0.

C.1.5.3 Digital Remote Procedure Call (DECrpc) – The Digital Remote Procedure Call (DECrpc) Version 1.0 is part of ULTRIX/UWS Version 4.0. The software is based on and is compatible with Version 1.5 of the Network Computing System (NCS) from the Apollo Systems Division of Hewlett-Packard Company. NCS is a set of tools for heterogeneous distributed computing.

Using remote procedure calls, software applications can be distributed across heterogeneous collections of computers, networks, and programming environments. Distributed applications can take advantage of computing resources throughout a network or Internet, with different parts of each program executing on the computers best suited for the tasks.

For more information about DECrpc, see the *DECrpc Programming Manual*, which provides programming information and examples for developing distributed applications. See also the *Guide to the Location Broker*, which describes `lb_admin`, the Location Broker administrative tool and the procedures for setting up and maintaining the local and global Location Broker daemons, `llbd` and `nrglbd`. The *ULTRIX Reference Pages* contain reference pages for each utility, special file, or library routine.

C.1.5.4 X/OPEN Transport Interface (XTI) – X/OPEN Transport Interface is a programming interface that conforms to the X/OPEN Standard for writing a portable network application. ULTRIX provides an XTI library that the network application can be linked with `(-lxti).` See `intro(3xti)` for more information.

C.1.5.5 Simple Network Management Protocol (SNMP) – Simple Network Management Protocol is the protocol used by the Internet (TCP/IP) for managing the network. Participating as an SNMP agent, ULTRIX can be managed by a local or remote network manager. See `snmpd(8n)` for more information.

ULTRIX/UWS Version 4.0 also allows you to write an Extended SNMP Agent for managing a private Management Information Base (MIB).

For additional information on writing an Extended SNMP Agent, see the *Guide to Network Programming* and `snmpext`(3n).

C.1.5.6 License Management Facility – The License Management Facility (LMF) has been added to this version of ULTRIX. The LMF enables the online management of software license data, and also helps prevent the accidental, unlicensed use of software.

With the introduction of the LMF come some important new features:

- Product Authorization Keys (PAKs)
  PAKs provide a way of encoding some of the software license information into a standard format that can be used by the LMF. PAKs are supplied by Digital when you order a product's license.
- The License Database (LDB)
  The LDB is a system file that contains the license data supplied by PAKs.
- The license management utility, `lmf`(8)
  The `lmf` utility is used to register and manage license data in the LDB.

A new manual, the *Guide to Software Licensing*, explains the new features introduced with the License Management Facility, and describes how to use the `lmf`(8) and the `lmfsetup`(8) utilities.

C.1.5.7 LMF and Capacity Upgrade Kit Distinctions – In ULTRIX/UWS Version 4.0, the functions of the Capacity Upgrade Kit are replaced by the License Management Facility (LMF).

ULTRIX/UWS Version 4.0 customers increase their user capacity using the License Management Facility shipped with ULTRIX/UWS Version 4.0 and documented in the *Guide to Software Licensing*.

Customers upgrading from earlier versions of the operating system can continue to increase their user capacity by using the Capacity Upgrade Kit they obtained with those earlier versions. Instructions for installing these Capacity Upgrade Kits were shipped with the earlier versions of ULTRIX as the document *ULTRIX Capacity Upgrade Installation Instructions*.

C.1.5.8 BSD curses and X/Open curses Libraries – ULTRIX/UWS Version 4.0 supplies two basic curses screen-handling software packages: BSD curses, and X/Open curses.

The BSD curses package consists primarily of the libcurses.a library and the `<curses.h>` header file. The ULTRIX/UWS Version 4.0 version is compatible with BSD 4.2, and with ULTRIX-32 Version 3.1, except that the `crmode()` and `nocrmode()` functions call the BSD 4.3 `cbreak()` and `nocbreak()` functions for forward compatibility with BSD 4.3 and backward compatibility with BSD 4.2. The BSD curses functions are summarized in the `curses(3x)` reference page.

The X/Open curses package consists primarily of the libcursesX.a library and the `<cursesX.h>` header file. The ULTRIX/UWS Version 4.0 version is conformant with the X/Open Portability Guide, Issue 3, except that the library name and the header file name contain the letter "X". This curses package is often referred to in ULTRIX documentation as "cursesX". The cursesX software is also compatible with AT&T System V Release 2, with AT&T SVID Issue 2, and with ULTRIX-32 Version 3.1. Section `3cur` of the *ULTRIX Reference Pages* contains a summary of the X/Open cursesX functions.

C.1.5.9 BSD curses Enhancements – The functions `cbreak()` and `nocbreak()` were added to the BSD curses library for forward compatibility with BSD 4.3. The functions `crmode()` and `nocrmode()` were redefined to call the new `cbreak()` and `nocbreak()` functions, respectively, to preserve backward compatibility with BSD 4.2 and ULTRIX-32 Version 3.1.

C.1.5.10 Changes in Terminfo Database – This note discusses changes to `terminfo` components.

C.1.5.10.1 The Binary Terminfo Database is Split into Two Subsets – The binary `terminfo` database (`/usr/lib/terminfo`) is now packaged in two subsets: a supported subset, and an unsupported subset. The supported subset is ULTBASE400 for VAX platforms, and UDTBASE400 for RISC platforms. The unsupported subset is ULXBASE400 for VAX platforms and UDXBASE400 for RISC platforms.

The supported components are those directories and files necessary to support Digital terminals and devices listed in the ULTRIX/UWS Version 4.0 Software Product Description (SPD). Everything else in the `terminfo` database is unsupported. The placement of `/usr/lib/terminfo` files into the supported and unsupported subsets is not necessarily an indicator of the support status of a particular directory or file. Certain unsupported components are placed in the supported subset in order to provide terminal-dependent functionality equivalent to functionalities supported by the Termcap database and `/usr/lib/term` database.

Other unsupported components are in the supported subset in order to provide ANSI compatibility, to provide backward compatibility with older ULTRIX releases, and to provide support for generic character-cell devices. Among those `terminfo` components which appear in the supported subset are files for support of the Teletype 37 terminal. The Teletype 37 is an unsupported terminal but files to support it are placed in the supported subset to provide backward compatibility and for functional equivalency.

The ULTRIX SPD lists the support status of a particular `terminfo` database component.

C.1.5.10.2 The Source Terminfo Database is Split into Two Subsets – The source files used to generate the ULTRIX/UWS Version 4.0 Terminfo Database files are located in `/usr/src/usr.lib/terminfo.` These source files are packaged in two subsets: a supported subset, and an unsupported subset. The supported subset is ULTPGMR400 for VAX platforms, and UDTPGMR400 for RISC platforms. The unsupported subset is ULXBASE400 for VAX platforms and UDXBASE400 for RISC platforms.

The supported components are those files necessary to support Digital terminals and devices listed in the ULTRIX/UWS Version 4.0 Software Product Description (SPD). Everything else in the source `terminfo` database is unsupported. The placement of `/usr/src/usr.lib/terminfo` files into the supported and unsupported subsets is not necessarily an indicator of the support status of a particular file. Certain unsupported files are placed in the supported subset in order to provide terminal-dependent functionality equivalent to functionalities supported by the Termcap database and `/usr/lib/term` database.

Other unsupported files are in the supported subset in order to provide ANSI compatibility, to provide backwards compatibility with older ULTRIX releases, and to provide support for generic character-cell devices. Among those source `terminfo` files which appear in the supported subset is the `teletype.ti` file which is for support of the Teletype 37 terminal. The Teletype 37 is an unsupported terminal, but the teletype.ti file is placed in the supported subset to provide backward compatibility and for functional equivalency.

All the source `terminfo` files in the supported subset are there because they are necessary to generate all the files in the binary `terminfo` files appearing in the supported ULTBASE400 and UDTBASE400 subsets. These source `terminfo` files contain information to generate `terminfo` binary files for some unsupported devices. Therefore, if the status of a `terminfo` source file is supported, this does not necessarily mean that the entire contents of that file are supported. Only those portions of a supported file which pertain to a supported device are supported.

The ULTRIX SPD lists the support status of a particular `terminfo` source file, or portion thereof.

C.1.5.10.3 Supported DEC Terminals Missing from the Terminfo Database – The Terminfo database does not currently provide support for the following supported DEC terminals:

- Specific models of the VT100 series except for models VT100, VT125, and VT132
- Specific models of the VT200 series

- Specific models of the VT300 series
- DECmate terminals
- DEC RAINBOW terminals
- DEC PRO 350 and 380 terminals
- VAXmate terminals
- Most DEC serial hardcopy terminals, except for the DECwriter series and generic printers

There are some possible workarounds for some of these terminals:

- You can set your terminal's terminal id (for all VT series terminals) to 'VT100 ID', 'VT200 ID', or 'VT300 ID'.
- You can set your TERM environment variable to 'vt100','vt200', or 'vt300' for VT series terminals. For more information, see `environ(7).`
- You can create alias entries in a supported `terminfo` source file, then compile the edited source using the `tic(1)` command. For more information, see `tic(1)` and `terminfo(5).`

A combination of the above workarounds is generally necessary for terminals which cannot be configured to act like a VT100, VT200 or VT300. The value for TERM should match your terminal's terminal id. You cannot set a value for TERM except to a value in the Termcap database. To use X/Open curses-based applications, the value of TERM must also correspond to a terminal type defined in the Terminfo database.

C.1.5.11 Changes to the Termcap Database – This section describes changes to the `termcap` database.

C.1.5.11.1 The Termcap Database Contains Unsupported Entries – Certain unsupported entries exist in `/etc/termcap` in order to provide terminal-dependent functionality equivalent to features supported by the Terminfo database and `/usr/lib/term` database. Other unsupported entries are in `/etc/termcap` in order to provide ANSI compatibility, to provide backwards compatibility with older ULTRIX releases, and to provide support for generic character-cell and communications devices. Among those Termcap entries are entries for the support of the Teletype 37 terminal. The Teletype 37 is an unsupported terminal but entries to support it exist to provide backward compatibility and for functional equivalency.

The ULTRIX SPD lists the support status of a particular Termcap entry.

C.1.5.11.2 Supported DEC Terminals Missing from the Termcap Database – The Termcap database does NOT currently provide support for the following supported DEC terminals:

- VT101 and VT102 terminals
- Specific models of the VT200 series
- Specific models of the VT300 series
- DECmate terminals
- DEC RAINBOW terminals

- DEC PRO 380 terminals
- VAXmate terminals
- Most DEC serial hardcopy terminals, except for the DECwriter series and "generic" printers

There are some possible workarounds for some of these terminals:

- You can set your terminal's terminal id (for all VT series terminals) to 'VT100 ID', or to 'VT200 ID', or to 'VT300 ID'.
- You can set your TERM environment variable to 'vt100', 'vt200' or 'vt300' for VT series terminals (see `environ(7)` for more information).
- You can create alias entries in a supported entry in `/etc/termcap`, or create a new entry. See `termcap(5)` for more information.

A combination of the above workarounds is generally necessary for terminals which cannot be configured to act like a VT100, VT200 or VT300. The value for TERM should match your terminal's terminal id. You cannot set a value for TERM except to a value in the Termcap database. To use BSD curses-based applications, the value of TERM must also correspond to a terminal type defined in the Termcap database.

C.1.5.12 Terminfo Terminal Capabilities Database Compiler and Sources – The `terminfo` database compiler, `tic(1)`, has been added to this version of the ULTRIX software. The `tic` compiler compiles terminal capabilities source files and updates the `terminfo` database, `/usr/lib/terminfo.` The source files used to generate the ULTRIX/UWS Version 4.0 version of `/usr/lib/terminfo` are also supplied in the `/usr/src/usr.lib/terminfo` directory. See `tic`(1) and `terminfo`(5) for more information.

C.1.5.13 Commands and Utilities – A limited number of commands and utilities have been modified for the following reasons:

- Bug fixing (described in Appendix A)
- Made 8-bit clean to meet X/Open compliancy
  - These commands are: `ar, awk, cat, cc, cd, chgrp, chmod, chown, cmp, comm, cp, cpio, date, diff, echo, /bin/echo, ed, egrep, expr, false, fgrep, find, gencat, grep, iconv, kill, lex, ln, lp, ls, mkdir, mv, pack, pcat, pg, pr, ps, pwd, /bin/pwd, red, rm, rmdir, sed, sh5, sleep, sort5, stty, tail, tar, tee, test, tr, true, tty, umask, uname, uniq, unpack, uucp, uulog, uuname, uupick, uustat, uuto, uux, wait, wc, who,` and `yacc`
  - In addition: `csh, sed, vi, ksh, tip, rlogin, more, head, adduser, login,` and `which`
- Adding BSD 4.3 enhancements: `cat, mv, plot, rmdir, w, chfn, chsh, finger, passwd, rdist, mkdir, chgrp, chown, cp,` and `ndbm`

- POSIX 1003.2, Draft 8 compliance: `basename, chgrp, chmod, chown, cp, date, fold, id, mkdir, mkfifo, ln, mktemp, sort, tee, tr, uniq, make, env, find, getopts, join, sh, ksh, awk, bc, cmp, diff,` and `dirname`.
- The ULTRIX/UWS Version 4.0 C Shell, `/usr/bin/csh,` based on the BSD 4.2 C Shell as distributed in ULTRIX-32, Version 3.0, features `vi` command line editing, active cursor keys and command completion. Unlike the BSD 4.3 C Shell, the ULTRIX C Shell does not edit the current command line.
- The ULTRIX `man` macros were enhanced as follows:
  - nroff output is centered inside an 80-column image area, and the line length increased to 70 characters, from 65
  - *troff output is for 8.5" x 11" pages, instead of 7" x 9"
  - The .TH macro syntax is now compatible with 4.3BSD
  - Only PostScript fonts are called when formatting with *troff

In addition, changes include improved compilers, specifically, pcc (VAX and RISC) and Vcc. Pcc was updated with a number of fixes as required by some SPRs which had been reported on the ULTRIX-32 Version 3.0 version. Vcc is described in the Section, VAX C/ ULTRIX.

C.1.5.14 Streaming Tape Devices and restore(8) – Prior to ULTRIX/UWS Version 4.0, `restore(8)` used standard synchronous I/O when reading from tape devices. Now, the command has been enhanced to use the n-buffered I/O facility. Thus, the command can stream a tape device such as the TU81 or TK50, reducing the time required to perform a restore of a file system.

C.1.5.15 Tape Exerciser, tapex(8) – A new tape exerciser, `tapex(8)`, has been added to the system. This utility provides a more comprehensive set of tests than that offered by `mtx(8).`

C.1.5.16 Caching Support for TMSCP Tape Driver – The TMSCP tape driver has been enhanced to provide caching support. Through the use of read-ahead and write-back caching, the throughput to certain TMSCP tape drives has been improved. To utilize TMSCP tape caching see `mtio(4)` for a description of the MTCACHE and MTFLUSH commands. The `dump(8)` utility has been modified to utilize caching on tape drives which support tape caching.

C.1.5.17 Configuration Support for 96 MSCP Disks – Support for MSCP (ra) disks has been enhanced to allow for configuration of up to 96 disks. Previously, support was only provided for 32 disks.

C.1.5.18 Exclusive Access Support for HSC Disks – The MSCP disk driver has been enhanced to provide exclusive access for disks connected to the HSC multi-host controller. This functionality provides for a higher degree of security on the CI by preventing other nodes on the CI from accessing the specified disk. See `radisk(8)` and `dkio(4)` for additional information.

C.1.5.19 New /sys Directory Structure – With ULTRIX/UWS Version 4.0 of ULTRIX, the directory structure for the `/sys` directory tree has changed. The directories used for configuring and building kernels have changed, but the process used to perform those tasks remains the same. Please consult the *Guide to Configuration File Maintenance* for detailed descriptions.

C.1.5.20 Tuning File System Performance – You can modify three system parameters to improve ULTRIX file system performance and demonstrate that improvement in certain types of tests, such as single-process, single-file, cache-sensitive benchmarks. However, before modifying these parameters, you should be knowledgeable about the ULTRIX operating system.

Depending on your system and its applications, you may possibly increase the file system performance by modifying the following three system parameters:

- Buffer cache size
- ULTRIX write-back scheduling
- The `update` daemon time interval

The following sections describe how to modify each of the parameters.

C.1.5.20.1 The bufcache Configuration File Parameter – A new configuration file parameter, *bufcache* , allows a specified percentage of physical memory to be set aside by the file system for use by the file system buffer cache. The percentage must be 10 or greater, but less than 100.

By default, buffer cache occupies 10% of main memory. Increasing the buffer cache size means that more file system data is stored in memory. While a large buffer cache may make a benchmark test run faster, there are tradeoffs. ULTRIX uses a static buffer cache allocation methodology. Main memory that is allocated at boot time for the file system buffer cache cannot be used for user program text or data. Therefore, actual performance depends on the application.

For example, to set the cache buffer size to 25% of memory, add the following to your system's configuration file located in the directory `/sys/conf/mips` for RISC processors or `/sys/conf/vax` for VAX processors:

```
bufcache              25
```

After editing the configuration file, you need to rebuild your kernel.

Optimal values for `bufcache` will differ among large timesharing systems, mid-range file servers, and workstations. However, you should not alter `bufcache` if you have a workstation with 8 Mbytes of memory. Workstations with 16 Mbytes of memory should have a value of no more than 30. If you specify a value greater than 30, your system's file system performance may suffer because of excessive paging and swapping.

For file servers, increasing the buffer cache can improve performance. Note that if you make the buffer cache too large, the resulting system may be less efficient in processing the requests to it from multiple users. To help you determine the optimal value, use the results from the `bufstats` command of the `crash` utility. This command can provide useful data on cache hit/miss ratios. See the `crash`(8) Reference Page for more information on `bufstats`.

See the *Guide to Configuration File Maintenance* for more information on the configuration file and its options and for instructions on rebuilding your kernel.

C.1.5.20.2 ULTRIX Write-Back Scheduling – By default, ULTRIX returns write requests immediately. If the last byte of a block is written, then the dirty block is asynchronously sent to disk. When this happens, the block becomes unavailable until the disk write completes. While this scheduling method is beneficial in a time-sharing environment, it hinders some benchmark tests which read data immediately after writing it.

To set the ULTRIX system so that data can be read as soon as it is written and writes to disk are delayed as long as possible, make the following change in the `param.c` file located in the directory `/sys/conf/mips` for RISC processors or `/sys/conf/vax` for VAX processors:

```
int delay_wbuffers = 1;
```

After editing the `param.c` file, you need to rebuild the kernel. See the *Guide to Configuration File Maintenance* for instructions on rebuilding your system's kernel.

C.1.5.20.3 The update Daemon Time Interval – By default, the `update` daemon synchronizes dirty blocks to disk every 30 seconds. You can alter this time interval in two ways. The first way is to add a value to the `/etc/update` command in the file `/etc/rc`. For example, to adjust the `update` time interval from 30 seconds to 2 minutes, edit the file as follows:

```
/etc/update 120;  echo -n update'  >/dev/console
```

The second way is to kill the `update` daemon process and restart it with the new value.

If you have a big cache and an application which often writes over the same blocks of a file, you should consider increasing the time interval for `update`. The judicious manipulation of buffer cache size, write scheduling strategy, and `update` frequency can improve file system performance. Individual users should analyze their needs by varying the values for each parameter and measuring the effect on performance. Optimal values will differ between workstations, file servers, and time-sharing systems.

Note

Unless you understand the value of modifying these parameters and can detect a performance improvement after doing so, you should use their default values.

C.1.5.21 PrintServer Client Software for ULTRIX – In Versions 3.0 and 3.1 of ULTRIX-32, print spooler support for the PrintServer family of network printers was supplied through a layered product, PrintServer Client Software for ULTRIX-32, Version 2.0.

In ULTRIX/UWS Version 4.0, this support has been incorporated into the base system, so you no longer need to install the PrintServer Client layered product.

All printer support is now contained in the subsets ULTPRINT400 (VAX) and UDTPRINT400 (RISC). You must install the appropriate subset before creating any print queues on your system.

Support is provided in ULTRIX for the following network printers:

- PrintServer 20
- PrintServer 40
- PrintServer 40 Plus

For management, booting, and file services you still need to install one of these separately-licensed products:

- PrintServer Supporting Host for ULTRIX (to boot and manage from ULTRIX systems)
- VAX PrintServer Supporting Host (to boot and manage from VMS systems).

For PrintServer network printers on a DECnet network, you require the product VMS PrintServer Supporting Host Software, Version 2.0, 2.1, or 3.0, running on a VAX system.

For PrintServer network printers on a TCP/IP network, you require the product PrintServer TCP/IP Supporting Host Software, Version 1.0 or 2.0, running on an ULTRIX VAX or RISC system.

C.1.5.22 New /etc/exports Semantics – The semantics of `/etc/exports` file options have been modified for ULTRIX/UWS Version 4.0. Export options are now applied on a per-directory basis and are no longer inherited from the exported parent filesystem. As a consequence, you no longer need to export an entire filesystem in order to export subdirectories within it. However, you do need to be more explicit in specifying options for each exported resource: filesystem or directory. See `exports`(5) for more information.

C.1.5.22.1 Differences and Benefits – The major difference is that desired export options need to be specified for each exported resource; they are no longer inherited from the exported parent file system.

The benefits are:

- Options now apply to directories; it is now possible, for instance, to set up a directory hierarchy such that higher level directories may be exported read-only, while some lower level directories may be exported read-write.
- Different exported directories in the same filesystem can now be given different export options (that is, rw, ro, rootmapping).
- There is no longer a need to export a filesystem to "nobody" so that individual directories can be rootmapped to uid 0 (typical in diskless clients).
- The "no file handle" option (–n) is no longer required.

Note that the access checking behavior differs from that of previous ULTRIX releases where access was checked in a top-down manner, starting with the filesystem. Previously, export options for the filesystem were in fact the options for any exported directory within the filesystem. In ULTRIX/UWS Version 4.0, access checking is actually applied bottom-up, as described below.

C.1.5.22.2 Preserving Current Export Behavior – The basic modification required for an existing `/etc/exports` file is that each exported resource identified in the file must now identify the export options associated with mounting that resource:

- If you want all exported directories within a filesystem to be exported read-only, you will need to specify the **-o** option on each `/etc/exports` entry.
- If you want to allow rootmapping to uid 0 (**-r=0**) then you will need to specify the **-r=0** option on each appropriate directory. You should delete `/etc/exports` entries for filesystem mount points that contain diskless root partitions since they typically look like:

```
/var/diskless -n -r=o nobody
```

- The **-r** option will now be specified on a per-directory basis and the **-n** option is no longer required. This option will not cause any harm if it remains, but it will be ignored in ULTRIX/UWS Version 4.0.

You may preserve the same behavior on your NFS server if you apply the export options of the exported parent filesystem to each of the exported subdirectories.

Note that a mount request for a subdirectory of an exported directory will be processed against the next highest exported directory identified in that hierarchy. See the next section for more detail.

C.1.5.22.3 Subtle Differences in Determining Access – A best match algorithm is used to determine which export options apply to a client NFS mount request. When a client makes an NFS mount request the export options that take effect are those of the exported directory specified in the server's / `etc/exports` file that most closely matches the request.

For example, if a client attempts to mount `/usr/man/man1` , and both `/usr/man` and `/usr/man/man1` are specified in the server's `/etc/exports` file, then access to the directory based upon the client's mount request is checked against the `/usr/man/man1` export access list first. If the mount is permitted, the directory options specified for `/usr/man/man1` are used. If, however, the client was not identified in the export access list (or in one of the groups specified) for `/usr/man/man1` , access is then checked against the closest exported ancestor, which in this case is `/usr/man`. If access is permitted at this level, the export options associated with `/usr/man` are used.

C.1.5.23 SCSI Drivers Support Dynamic Bad Block Replacement (DBBR). – The SCSI subsystem, the DECstation 2100, DECstation 3100, MicroVAX 3100, DECsystem 3100, VAXstation 3100, VAXstation 3520, VAXstation 3540, DECsystem 5000 and the DECstation 5000 now support Dynamic Bad Block Replacement (DBBR). The SCSI drivers will automatically replace soft errors that are ECC correctable. The user is notified of the replacement by the use of the error log file.

Hard disk errors are still reported and logged by the error logger. There is no attempt to reassign hard errors; this requires user intervention.

C.1.5.24 SCSI Driver Logs Errors in Binary – The SCSI driver no longer logs its errors in ASCII. Instead extensive changes have been made to log errors in binary form allowing the `uerf` utility to improve the amount of error information presented in the error log.

C.1.5.25 Time Zone Handling – The time zone related library routines have been enhanced to handle a wider range of time zone rules and to conform to the POSIX P1003.1 standard.

The TZ environment variable can be set by users to tailor time zone handling.

The standard defines the format of the TZ string such that only one time zone rule can be defined at a time; different rules for previous years are not handled.

The standard also allows implementation defined rules. ULTRIX/UWS Version 4.0 takes advantage of this feature of the standard by allowing the TZ string to specify a data file which contains the rules for a particular time zone.

The data files, which are located in `/etc/zoneinfo`, are created by the zone information compiler (zic), from the source files located in `/etc/zoneinfo/sources`.

If no TZ string is defined, the rules specified by the file `/etc/zoneinfo/localtime` are used. This file is set up based on information supplied to the installation procedure. If the created file does not accurately represent the rules for your time zone, you can use the `zic(8)` command to create a new file.

See the `ctime(3)` and `zic(8)` reference pages for more detailed information on the format and use of the TZ environment variable and the time zone data files.

C.1.5.26 Support for Multiple Databases – The previous version of `dbm(3x)` only allows one database to be used at a time. A new library, `ndbm`(3) is now provided as part of the libc library to augment the `dbm` functionality. Now, `ndbm` allows multiple databases to be concurrently accessed. Existing `dbm` calls can either be replaced by `ndbm` calls or they may continue to use `dbm`, which has remained in the `libdbm` library.

C.1.5.27 New Features for Writing International Software – This version of the ULTRIX system adds support for internationalization features to three functions and adds a new internationalization command. This version also supports the use of *@modifier* in *locale* settings.

The `scanf`, `printf`, and `vprintf` functions now contain support for internationalization features. These internationalized functions now give you the flexibility to format the same data in different ways to accommodate users in different international areas. You can use the internationalization features if you link your program with the internationalization library, *libi*.

The behavior of the internationalized routines is different from the behavior of the *libc* routines `printf`(3s), `scanf`(3s), and `vprintf`(3s).

For more information on using the internationalization features of the `scanf`, `printf`, and `vprintf` functions, see the `scanf`(3int), `printf`(3int), and `vprintf`(3int) reference pages.

Note

The `scanf`, `printf`, and `vprintf` internationalized functions supercede the `nl_scanf` and `nl_printf` functions, which are still available on the system for conformance with the X/Open XPG–2 standard. However, you should avoid using the superceded functions unless you must conform to that standard, because there is a possibility the functions will not be supported in future releases of ULTRIX.

The new internationalization command is the `iconv` command. This command allows you to convert the encoding of data from one codeset to another codeset. You control how `iconv` converts the data. For more information on the new command, see the `iconv`(1) reference page.

The behavior of the `ic` compiler and the `setlocal` function have been changed to support the @*modifier* in *locale* settings. This feature allows you to specify more detail for the settings. For example, you can use @*modifier* to load a table that specifies telephone or dictionary collation of data, as opposed to the default collation for a particular locale. See the *Guide to Developing International Software*, the `setlocale`(3int), and the `lang`(5int) reference pages for more information.

Note

Due to the nature of the changes, we recommend you recompile any language support databases using the new version of the `ic` compiler. In addition, you should relink programs that had previously used `setlocale` with `libc`.

C.1.5.28 Security Enhancements – ULTRIX security has been enhanced by several new features.

User authentication features now support the following:

- Storage of encrypted passwords in an authentication data base readable only by root, rather than in `/etc/passwd`.
- Password aging and expiration.
- Support of passwords up to 16 characters long.
- Enforced minimum required password length.
- Automatic generation of passwords that are difficult to guess.

The system administrator has the option of using the traditional authentication system, the new system, or a mode designed to allow easy transition from the old authentication system to the new one.

A new audit subsystem is capable of recording a wide range of events and logging the information in a secure audit log file. The system administrator can choose a list of events to log for all users and then add or remove events from that list on a per-user basis to tailor the logging to individual users. A special tool enables filtering of the audit file and produces reports focusing on needed information. The new audit subsystem can co-exist with the traditional ULTRIX accounting features.

A new trusted path feature provides the user with a secure path between the user's terminal and the legitimate login process. This prevents an illicit program from faking the login process and grabbing the user's password.

A security setup command is available to simplify the task of configuring the new system security features.

C.1.5.29 Symmetric Multiprocessing (SMP) – For ULTRIX/UWS Version 4.0, the kernel has been modified to allow multiple processors to execute the kernel code simultaneously. This is accomplished safely by means of locks, which are used to

control the concurrent access of shared data structures within the kernel. ULTRIX/UWS Version 4.0 now supports the entire line of VAX and RISC multiprocessor systems, excluding the VAX 11/782.

Commands added for symmetric multiprocessing are:

- `startcpu(8)`, which controls starting an attached processor.
- `stopcpu(8)`, which controls stopping an attached processor. Stopping the boot processor is not allowed.
- `cpustat(1)`, which prints out CPU usage statistics on a per-processor basis.

Symmetric multiprocessing is enabled by a kernel configuration file option "SMP". Without this option in the system configuration file, attached processors can not be started. With this option, up to the supported maximum number of processors can be started.

While upgrading any system with only one cpu to one with multiple cpus, you must edit the system configuration file of the machine to define SMP as an option. Include this line in the system configuration file:

```
options SMP
```

After you change the system configuration file, be sure to run the `config`(8) program on it and then run make depend and make vmunix for the new system configuration.

Not all device drivers supplied by Digital have been made symmetric. Those that have not been modified are still supported by asymmetric driver support. Whether a device driver is symmetric or not is determined by the field "d_affinity" in the block device switch (`bdevsw`) or character device switch (`cdevsw`). A value of zero means that only the boot processor can execute the driver. A value of -1 means that all processors can execute the driver.

C.1.5.30 VAX C/ULTRIX – This section of the release notes covers new features added in ULTRIX/UWS Version 4.0 of VAX C/ULTRIX (vcc). VAX C/ULTRIX is for VAX machines only. There is also new documentation for VAX C/ULTRIX included in ULTRIX/UWS Version 4.0. You should consult the *VAX/C for ULTRIX* documentation for details on new features.

C.1.5.30.1 New Object Format – In ULTRIX/UWS Version 4.0 of VAX C/ULTRIX, the vcc compiler now generates BSD .o format for its object files. This means that the ld linker can now be used to link object files generated by vcc.

The previous object file format can still be generated, but this is controlled by a new qualifier, -V LKOBJECT. The default value of this switch is -V NOLKOBJECT.

The vcc shell will automatically pass the BSD objects to ld to be linked. If files with a .obj extension appear on the command line, or if -V LKOBJECT is specified, the files will be passed to lk instead.

You should note that the object files produced by both -V LKOBJECT and -V NOLKOBJECT are both suffixed by the .o file extension. However, ld will not link files produced by using vcc -V LKOBJECT or produced by versions of vcc prior to ULTRIX/UWS Version 4.0. If you attempt to do so, the ld linker will tell you that the symbol _MUST_USE_LK_TO_LINK_THIS_OBJECT is undefined.

C.1.5.30.2 Function Inlining – In ULTRIX/UWS Version 4.0 of VAX C/ULTRIX, a significant new optimization has been implemented called function inlining. In this optimization the compiler automatically attempts to place the code of a function body directly inline at the locations where the function is called. This eliminates the overhead of a call instruction. This optimization is only attempted for functions defined in the module.

Not all functions are automatically inlined. The compiler estimates the code size and amount of CPU time that the function consumes, and uses these heuristics to select candidates for automatic inlining. Functions will not be automatically inlined if the compiler judges that performance would not be significantly improved. However, you are able to override the compiler's decision on any individual eligible function by specifying the function in a pragma:

```
#pragma [no]inline (function_name [,function_name...])
```

The [no]inline pragma must appear before the definitions of the functions named. However, the definition of a function need not precede its calls for the function to be inlined.

Some functions cannot be inlined at all. Functions that take the address of their parameters, or functions that use varargs are not inlined by the compiler. If the compiler is unable to inline a function that you have suggested be inlined (with #pragma inline), informational messages are generated.

You can also specify that the optimizer should run but not perform function inlining by using the qualifier -V OPTIMIZE=NOINLINE on the command line. The default is -V OPTIMIZE=INLINE.

C.1.5.30.3 Access to Specialized VAX Instructions – In ULTRIX/UWS Version 4.0 of VAX C/ULTRIX, access to some VAX instructions is possible directly from C code. In particular, access to the following instructions is provided: ADAWI, BBCCI, BBSSI, FFC, FFS, HALT, INSQHI, INSQTI, INSQUE, LDPCTX, LOCC, MFPR, MOVC3, MOVC5, MOVPSL, MTPR, PROBER, PROBEW, REMQHI, REMQTI, REMQUE, SCANC, SKPC, SPANC, and SVPCTX.

These instructions are accessed through a function call syntax. These functions give you capabilities similar to asm in pcc. However, instead of providing a string which contains an assembler instruction as the parameter as asm requires, C variables and expressions are the parameters.

A new pragma has been added to ULTRIX/UWS Version 4.0 of VAX C/ULTRIX to allow you to selectively enable or disable compiler recognition of these instruction functions. The syntax of the new pragma is:

```
#pragma [no]builtins
```

The default setting is #pragma nobuiltins. You must include the builtins pragma in your modules prior to calls to builtin functions. See the *Guide to VAX C/ULTRIX* for information on the parameters to these builtin functions.

C.1.5.30.4 New Behavior for -E – In ULTRIX/UWS Version 4.0 of VAX C/ULTRIX, cpp is no longer invoked when -E is specified on the command line. Instead, VAX C/ULTRIX preprocesses the file, and produces source output. There is not a separate preprocessor; this capability is built into the VAX C/ULTRIX compiler.

The use of the -Em qualifier will still cause cpp to be invoked. VAX C/ULTRIX does not generate makefile dependencies.

C.1.5.30.5 Function Pointer Syntax Now Accepted – In ULTRIX/UWS Version 4.0 of VAX C/ULTRIX, function pointers are now allowed to be used in function calls without being deferenced, as pcc allows. Thus, if p is a pointer to a function, the following expressions are equivalent:

```
(*p)()
```

and

```
p()
```

These expressions are calls to the function pointed at by p. VAX C/ULTRIX will accept both expressions as legal.

C.1.5.30.6 Minor ANSI C Extensions – In ULTRIX/UWS Version 4.0 of VAX C/ULTRIX, floating point constants can take the following suffixes, as allowed by the draft ANSI C standard:

Lowercase "l"
Uppercase "L"
Lowercase "f"
Uppercase "F"

If the constant has the suffix "l" or "L", then it has type double. If the constant has the suffix "f" or "F", then it has type float. Unsuffixed floating point constants continue to be interpreted as having type double.

If an integer constant has the uppercase "U" or lowercase "u" suffix, it is an unsigned constant. The "U" or "u" suffix can be used in conjunction with the "l" or "L" suffix to create constants of type unsigned long.

C.1.5.30.7 Pragma for enabling/disabling -V STANDARD=PORTABLE – In ULTRIX/UWS Version 4.0 of VAX C/ULTRIX, a new pragma has been implemented which allows you to selectively turn off the -V STANDARD=PORTABLE qualifier for sections of your code.

The syntax for this new qualifier is:

```
#pragma [no]standard
```

If #pragma nostandard is specified, it turns off -V STANDARD=PORTABLE checking; this pragma has no effect if -V STANDARD=PORTABLE has not been specified on the command line. If #pragma standard is specified, it allows the previous setting of -V STANDARD=PORTABLE to be in effect. These pragmas can be nested; if two nostandard pragmas are specified in a row, two standard pragmas are required to restore portability checking.

C.1.6 Documentation Component Features

There have been many changes and additions to the documentation for ULTRIX/UWS Version 4.0. Please refer to the manual *Reader's Guide and Master Index* for a thorough introduction to the documentation.

C.1.7 Reference Pages Subsections Defined

The list of subsections is important because layered products can only use subsections known to `/usr/ucb/man`. For example, someone developing a layered product called XYZ wants all their reference pages to be in `xyz` subsections. Because `xyz` is not known to `/usr/ucb/man` (the default `man` command), the `alpha.1xyz` reference page will not be found by typing the following command:

```
% man alpha
```

The only way to find the `alpha.1xyz` reference page is to type one of the following commands:

```
% man 1xyz alpha

% /usr/bin/man alpha
```

The list of subsections known to `/usr/ucb/man` is hard coded, and is generally expanded for each ULTRIX release. Therefore, you need to use `/usr/bin/man` if `/usr/ucb/man` can't find a reference page. See the `man`(1) reference page for more information.

One of the features of `/usr/ucb/man` is that it supports subsections. The reference pages database is divided up into 8 standard sections (1-8), plus nonstandard sections l, n, o, p, 0, and 9. Six of the standard sections can be further split into subsections. Each subsection is a section suffix consisting of 1 or more alphanumeric characters, the first of which must be a letter. Sections 1-5 and 8 can have reference pages assigned to subsections.

The command `/usr/ucb/man` has a restriction in that it recognizes a hard-coded list of subsections. If a reference page is added to a subsection not known to `/usr/ucb/man`, that reference page will not be found unless the subsection is specifically requested. Therefore, it is important to know what subsections are currently defined, and the topics associated with those subsections.

The following is a list of the subsections defined for the ULTRIX/UWS Version 4.0 release. In some cases the meaning of a subsection depends on the primary section number.

```
Subsection    Topic
----------    ------------------------------------------------

 c       - Section 1: commands for communication with other systems
           Section 2: *** No Defined Meaning ***
           Section 3: subroutines for communication with other systems
           Section 4: *** No Defined Meaning ***
           Section 5: *** No Defined Meaning ***
           Section 8: commands for communication with other systems

 cur     - X/Open curses
 dn      - DECnet-ULTRIX (Layered Product)
 Dwt     - ULTRIX Worksystem Software (UWS) DECwindows XUI Toolkit
              Functions

 f       - Section 1: *** No Defined Meaning ***
           Section 2: *** No Defined Meaning ***
           Section 3: FORTRAN* reference pages
           Section 4: network protocol families
              Section 5: *** No Defined Meaning ***
           Section 8: *** No Defined Meaning ***

 g       - Section 1: commands used for graphics & computer-aided
```

```
            design, plus DECPhigs/ULTRIX (Layered Product)
          Section 2: *** No Defined Meaning ***
          Section 3: DECPhigs/ULTRIX (Layered Product)
          Section 4: *** No Defined Meaning ***
          Section 5: *** No Defined Meaning ***
          Section 8: *** No Defined Meaning ***

int     - Internationalization commands and functions

j       - *** No Defined Meaning ***
            - supported by BSD 4.2 /usr/ucb/man but topic is not
              known

krb     - Kerberos

m       - Section 1: *** No Defined Meaning ***
          Section 2: *** No Defined Meaning ***
          Section 3: Math Subroutines
          Section 4: *** No Defined Meaning ***
          Section 5: *** No Defined Meaning ***
          Section 8: System Maintenance

mh      - RAND Mail Handler (MH)

n       - Section 1: *** No Defined Meaning ***
          Section 2: *** No Defined Meaning ***
          Section 3: Network subroutines and SNMP
          Section 4: networking facilities
          Section 5: Simple Network Management Protocol (SNMP)
          Section 8: Simple Network Management Protocol (SNMP)

ncs     - Network Computing System (NCS)
nfs     - Networked File System (NFS)

osi     - DECnet-ULTRIX (Layered Product) Open Systems Interface (OSI)

p       - Section 1: *** No Defined Meaning ***
          Section 2: *** No Defined Meaning ***
          Section 3: *** No Defined Meaning ***
          Section 4: Internet Protocols
          Section 5: *** No Defined Meaning ***
          Section 8: *** No Defined Meaning ***

r       - *** No Defined Meaning ***
            - supported by BSD 4.2 /usr/ucb/man but topic is not
              known

s       - Section 1: *** No Defined Meaning ***
          Section 2: *** No Defined Meaning ***
          Section 3: Standard I/O Subroutines
          Section 4: *** No Defined Meaning ***
          Section 5: *** No Defined Meaning ***
          Section 8: *** No Defined Meaning ***

sh5     - System V Shell

sql     - ULTRIX/SQL components

svs     - *** RESERVED FOR FUTURE USE BY Digital Equipment Corp. ***

ufs     - ULTRIX File System (UFS)

v       - Sections 1-5: *** No Defined Meaning ***
          Section 8: Archiver, "crash", and disk format
```

```
X       - UWS X Graphic applications, Graphic maintenance commands, X
                servers and X system utilities

x       - Section 3: Special Library Functions
             - miscellaneous subroutines and libraries

X11     - UWS X11 Xlib Functions

Xt      - UWS DECwindows XUI Toolkit Intrinsics

xti     - X/Open Transport Interface

yp      - Yellow Pages (YP)
```

New subsections may be added in future releases.

The eight standard sections correspond to 8 subdirectories in `/usr/man`: man1 - man8. These sections have the following meanings:

```
1    - Commands
2    - System Calls
3    - Subroutines
4    - Special Files
5    - File Formats
6    - Games
7    - Macro Packages and Conventions (plus miscellaneous)
8    - Maintenance
```

In addition to the eight standard sections, customers may create the following additional subdirectories in `/usr/man`:

```
manl      - for "local" reference pages
mann      - for "new" reference pages
mano      - for "old" reference pages
manp      - for "public" reference pages
man0      - *** No Defined Meaning ***
man9      - *** No Defined Meaning ***
```

The files in `/usr/man/manl` must have names ending in .l, files in mann must names ending in .n, and so forth.

C.1.8 Customer Services Components Features

System integrated and individual software services are available for ULTRIX. Individual services include telephone support, installation services, and media and documentation update services. For more information on these and new services for ULTRIX, contact your local sales office.

In addition, three services have been added:

- System Management Service (SMS)

  SMS provides single point of access to a customer support center and a proactive problem resolution process.

- Software Update Installation Service (SUIS)

  SUIS provides a software update installation at a customer site by Digital specialists, who also explain the changes in the new release of the software.

- Source Code Update Service (SCUS)

  SCUS provides automatic updates of source code to customers with each scheduled version release of the operating system.

C.1.9 Software Features No Longer Supported

All software components supported in the previous release (ULTRIX-32, Version 3.1) remain supported in ULTRIX/UWS Version 4.0.

C.1.10 Hardware No Longer Supported

All hardware components supported in the previous release (ULTRIX-32, Version 3.1) remain supported in ULTRIX/UWS Version 4.0.

C.2 ULTRIX Worksystem Software Changes and New Features

This section discusses the following changes and new features in ULTRIX Worksystem Software that apply to ULTRIX/UWS Version 4.0:

- Conformance to Standards
- X Window System Changes
- DECwindows Toolkit Changes
- DECwindows Application Changes
- Font Format Changes
- Conformance to Standards

C.2.1 X Window System

This section discusses X Window System-related changes in ULTRIX/UWS Version 4.0

C.2.1.1 Xlib Changes – Changes to the Xlib data structures and programming interfaces to support the Inter-Client Communication Conventions include:

- Removal of the `PAllWMSizeHints` macro.
- Removal of the `XGetWMCommand` and `XSetWMCommand` routines.
- Correction of the order of screen and window arguments in `XIconifyWindow`, `XWithdrawWindow`, and `XReconfigureWMWindow`.
- The `killid` field in the definition of `XStandardColormap` has not been voted on by the wmtalk (window manager talk) list but is viewed as a necessary change by the Director of the X Consortium. A change in this part of the interface is possible if the wmtalk group comes up with an alternative proposal.

C.2.1.2 New Xlib Programming Interfaces – In ULTRIX/UWS Version 2.2, the following interfaces were added to Xlib.

C.2.1.2.1 Allocating Structures for Property Data – The following routines allocate memory for the various data structures reflecting information stored in properties. By allocating these structures at run time instead of declaring them at compile time, clients avoid memory overwriting problems should additional fields ever be added to these structures.

```
XSizeHints *XAllocSizeHints ()
```

```
XStandardColormap *XAllocStandardColormap ()
XWMHints *XAllocWMHints ()
XClassHint *XAllocClassHint ()
XIconSize *XAllocIconSize ()
```

The XAllocSizeHints, XAllocStandardColormap, XAllocWMHints, XAllocClassHint, and XAllocIconSize routines allocate and return XIconSize XAllocIconSize () pointers to XSizeHints, XStandardColormap, XWMHints, CWXClassHint, and XIconSize structures, respectively. Pointer fields will be set to NULL and all other fields will be set to zero. If sufficient memory is not available, NULL is returned.

C.2.1.2.2 Manipulating Top-Level Windows – The following routines change the size or visibility of top-level windows (created as children of the root window).

The XIconifyWindow function sends a WM_CHANGE_STATE ClientMessage event with a format of 32 and a first data element of IconicState to the root window of the specified screen (for more information on IconicState, see the *Inter-Client Communications Manual*, Section 4.1.4, Changing Window State) Window managers may elect to receive this message and, if the window is in its normal state, may treat it as a request to change the window's state from normal to iconic. If the WM_CHANGE_STATE atom cannot be interned, no message is sent, and a status of 0 is returned. A nonzero status is returned if the client message is sent successfully; otherwise, a status of 0 is returned.

```
Status XIconifyWindow (dpy, w, screen)
    Display *dpy;
    Window w;
    int screen;
```

The XIconifyWindow function can generate a BadWindow error.

The XWithdrawWindow function unmaps the specified window and sends a synthetic UnmapNotify event to the root window of the specified screen. Window managers may elect to receive this message and treat it as a request to change the window's state to withdrawn. When a window is in the withdrawn state, neither its normal nor its iconic representations is visible. A nonzero status is returned if the UnmapNotify event is sent successfully; otherwise, a status of 0 is returned.

```
Status XWithdrawWindow (dpy, w, screen)
    Display *dpy;
    Window w;
    int screen;
```

The XWithdrawWindow function can generate a BadWindow error.

The XReconfigureWMWindow function does a ConfigureWindow on the specified top-level window. If the stacking mode is changed and the request fails with a BadMatch error, the error event is trapped and a synthetic ConfigureRequestEvent containing the same configuration parameters is sent to the root of the specified window. Window managers may elect to receive this event and treat it as a request to reconfigure the indicated window.

```
Status XReconfigureWMWindow (dpy, w, screen, mask, changes)
    Display *dpy;
    Window w;
    int screen;
    unsigned int mask;
    XWindowChanges *changes;
```

The XReconfigureWMWindow function can generate BadValue and BadWindow errors.

C.2.1.2.3 String Lists – The following routines convert between lists of pointers to character strings and text properties.

The XStringsToTextProperty routine sets the specified XTextProperty to be of type STRING (format 8) with a value representing the concatenation of the specified list of null-separated character strings. An extra byte containing NULL (which is not included in the count tally) is allocated for the value field of text_prop_return. Storage for this field may be released using XFree. If insufficient memory is available for the new value string, none of the fields in text_prop_return are set and 0 is returned; otherwise, a nonzero status is returned.

```
Status XStringListToTextProperty (list, count, text_prop_return)
    char **list;
    int count;
    XTextProperty *text_prop_return;
```

The XTextPropertyToStringList routine returns a list of strings representing the null-separated elements of text_prop. The data in text_prop must be of type STRING and format 8; multiple elements (such as the strings in a disjoint text selection) are separated by a NULL (encoding 0). The property is not null-terminated. Storage for the list and its contents may be released using XFreeStringList. If insufficient memory is available for the list and its elements, neither of the return values is set and a status of 0 is returned; otherwise, a nonzero status is returned.

```
Status XTextPropertyToStringList (text_prop, list_return, count_return)
    XTextProperty *text_prop;
    char ***list_return;
    int *count_return;

void XFreeStringList (list)
    char **list;
```

The XFreeStringList routine releases memory allocated by XTextPropertyToStringList.

C.2.1.2.4 Manipulating Text Properties – The following routines are used to obtain and specify information on properties that are intended to hold text. Note that Xlib does not enforce any encoding of textual information.

The XGetTextProperty routine reads the specified property from the window and stores the data in the value field of text_prop_return, the type of the data in the encoding field, the format of the data in the format field, and the number of items of data in the nitems field. The particular interpretation of the property's encoding and data as text is left to the calling application.

```
Status XGetTextProperty (display, window, text_prop_return, property)
    Display *display;
    Window window;
    XTextProperty *text_prop_return;
    Atom property;
```

If the property does not exist on the window, the value field of text_prop_return is set to NULL, the encoding field is set to None, the format field is set to 0, and the nitems field is set to 0.

The function returns a nonzero status if it was able to set the fields of text_prop_return; otherwise it returns 0.

The `XGetTextProperty` routine can generate BadWindow, BadAtom, or BadValue errors.

The `XGetWMName`,`XGetWMIconName`, and `XGetWMClientMachine` routines are convenience routines that perform a XGetTextProperty on the WM_NAME, WM_ICON_NAME, WM_CLIENT_MACHINE properties respectively.

```
Status XGetWMName (dpy, w, text_prop_return)
Status XGetWMIconName (dpy, w, text_prop_return)
Status XGetWMClientMachine (dpy, w, text_prop_return)
    Display *dpy;
    Window w;
    XTextProperty *text_prop_return;
```

The `XGetWMName` routine supersedes `XFetchName`. The `XGetWMIconName` routine supersedes `XGetIconName`.

The `XSetTextProperty` routine replaces (or sets if the property does not exist) the specified property for the indicated window with the data given by the value field of text_prop, the type given by the encoding field, the format given by the format field, and the number of items given by the nitems field.

```
void XSetTextProperty (dpy, w, text_prop, property)
    Display *dpy;
    Window w;
    XTextProperty *text_prop;
    Atom property;
```

The `XSetTextProperty` routine can generate BadWindow, BadAtom, BadValue, or BadAlloc errors.

The `XSetWMName`, `XSetWMIconName`, and `XSetWMClientMachine` routines are convenience routines that perform an `XSetTextProperty` on the WM_NAME, WM_ICON_NAME, and WM_CLIENT_MACHINE properties, respectively.

```
void XSetWMName (dpy, w, text_prop)
void XSetWMIconName (dpy, w, text_prop)
void XSetWMClientMachine (dpy, w, text_prop)
    Display *dpy;
    Window w;
    XTextProperty *text_prop;
```

The `XSetWMName` routine supersedes `XStoreName`. The `XSetWMIconName` routine supersedes `XSetIconName`.

The `XGetCommand` routine reads the WM_COMMAND property from the specified window and returns a string list. If the WM_COMMAND property exists, is of type STRING and format 8, and if there is enough memory to contain the string list, the `argvp` and `argcp` fields are filled in with a string list that may be freed with `XFreeStringList`. Upon successful completion, a nonzero status is returned; otherwise, a 0 status is returned.

```
Status XGetCommand (dpy, w, argvp, argcp)
    Display *dpy;
    Window w;
    char ***argvp;
    int *argcp;
```

C.2.1.2.5 Size Hints – The following routines obtain and specify size hints:

The `XGetWMSizeHints` routine returns the size hints stored in the indicated property on the specified window. If the property is of type WM_SIZE_HINTS and format 32 and is long enough to contain a size hints structure, the various fields of the hints_return structure are set and a nonzero status is returned; otherwise, a status of 0 is returned. To get a window's normal size hints, the `XGetWMNormalHints` routine may be used instead.

```
Status XGetWMSizeHints (dpy, w, hints_return, property)
    Display *dpy;
    Window w;
    XSizeHints *hints_return;
    Atom property;
```

The `XGetWMSizeHints` routine can generate BadWindow, BadAtom, or BadValue errors.

The `XGetWMSizeHints` routine supersedes `XGetSizeHints`.

The `XSetWMSizeHints` routine replaces (or sets if the property does not exist) the size hints for indicated property on the specified window. The property is stored with a type of WM_SIZE_HINTS and a format of 32. To set a window's normal size hints, the `XSetWMNormalHints` routine may be used instead.

```
void XSetWMSizeHints (dpy, w, hints, property)
    Display *dpy;
    Window w;
    XSizeHints *hints;
    Atom property;
```

The `XSetWMSizeHints` routine can generate BadWindow, BadAtom, or BadValue errors.

The `XSetWMSizeHints` routine supersedes `XSetSizeHints`.

The `XGetWMNormalHints` routine returns the size hints stored in the WM_NORMAL_HINTS property on the specified window. If the property is of type WM_SIZE_HINTS, format 32, and is long enough to contain a size hints structure, the various fields of the hints_return structure are set and a nonzero status is returned; otherwise, a status of 0 is returned.

```
Status XGetWMNormalHints (dpy, w, hints_return)
    Display *dpy;
    Window w;
    XSizeHints *hints_return;
```

The `XGetWMNormalHints` routine can generate BadWindow, BadAtom, or BadValue errors.

The `XGetWMNormalHints` routine supersedes `XGetNormalHints`.

The `XSetWMNormalHints` routine replaces (or sets if the property does not exist) the size hints for the WM_NORMAL_HINTS property on the specified window. The property is stored with a type of WM_SIZE_HINTS and a format of 32.

```
void XSetWMNormalHints (dpy, w, hints)
    Display *dpy;
    Window w;
    XSizeHints *hints;
```

The `XSetWMNormalHints` routine can generate BadWindow, BadAtom, or BadValue errors.

The XSetWMNormalHints routine supersedes XSetNormalHints.

C.2.1.2.6 Window Manager Protocols List – The following routines obtain and specify the list of window manager protocols in which the client is willing to participate:

The XGetWMProtocols routine returns the list of atoms stored in the WM_PROTOCOLS property on the specified window. These atoms describe window manager protocols in which the owner of this window is willing to participate. If the property exists, is of type ATOM, is of format 32, and the atom WM_PROTOCOLS can be interned, protocols_return is set to a list that the caller may release with XFree of atoms, and count_return is set to the number of elements in the list. Upon successful completion, a nonzero status is returned; otherwise, a status of 0 is returned and neither of the return values is set.

```
Status XGetWMProtocols (dpy, w, protocols_return, count_return)
    Display *dpy;
    Window w;
    Atom **protocols_return;
    int *count_return;
```

The XGetWMProtocols routine can generate BadWindow, BadAtom, or BadValue errors.

The XSetWMProtocols routine replaces (or sets if the property does not exist) the WM_PROTOCOLS property on the specified window to contain the list of atoms given by protocols. The property is stored with a type of ATOM and a format of 32. If the routine is unable to intern the atom WM_PROTOCOLS, a status of 0 will be returned. Otherwise, a nonzero status is returned.

```
Status XSetWMProtocols (dpy, w, protocols, count)
    Display *dpy;
    Window w;
    Atom *protocols;
    int count;
```

The XSetWMProtocols routines can generate BadWindow, BadAtom, BadValue, or BadAlloc errors.

C.2.1.2.7 Window Manager Colormap Windows List – The following routines are used to obtain and specify the list of windows whose colormaps should be installed by the window manager:

The XGetWMColormapWindow routine returns the list of window identifiers stored in the WM_COLORMAP_WINDOWS property on the specified window. These windows indicate the colormaps that the window manager may need to install for this window. If the property exists, is of type WINDOW, is of format 32, and the atom WM_COLORMAP_WINDOWS can be interned, windows_return is set to a list that the caller may release with XFree of window identifiers, and count_return is set to the number of elements in the list. Upon successful completion, a nonzero status is returned; otherwise, a status of 0 is returned and neither of the return values is set.

```
Status XGetWMColormapWindow (dpy, w, windows_return, count_return)
    Display *dpy;
    Window w;
    Window **windows_return;
    int *count_return;
```

The XGetWMColormapWindows routine can generate BadWindow, BadAtom, or BadValue errors.

The XSetWMColormapWindows routine replaces (or sets if the property does not exist) the WM_COLORMAP_WINDOWS property on the specified window to contain the list of windows given by colormap_windows. The property is stored with a type of WINDOW and a format of 32. If the routine is unable to intern the atom WM_COLORMAP_WINDOWS, a status of 0 will be returned; otherwise, a nonzero status is returned.

```
Status XSetWMColormapWindows (dpy, w, colormap_windows, count)
    Display *dpy;
    Window w;
    Windows *colormap_windows;
    int count;
```

The XSetWMProtocols routine can generate BadAlloc, BadAtom, and BadWindow errors.

C.2.1.2.8 Standard Colormaps – The following functions are used to obtain and specify standard colormaps:

The XGetRGBColormaps routine returns the RGB colormap definitions stored in the indicated property on the specified window. If the property exists, is of type RGB_COLOR_MAP, is of format 32, and is long enough to contain a colormap definition (if the visualid is not present, the default visual for the screen on which the window is located is assumed; if the killid is not present, None, indicating that the resources cannot be released, is assumed), space for the returned colormaps is allocated and is filled in. Upon successful completion, a nonzero status is returned; otherwise, none of the fields are set and a status of 0 is returned. It is the caller's responsibility to honor the *Inter-Client Communications Conventions Manual* (ICCCM) restriction that only RGB_DEFAULT_MAP contain more than one definition.

```
Status XGetRGBColormaps (dpy, w, stdcmap_return, count_return, property)
     Display *dpy;
     Window w;
     XStandardColormap **stdcmap_return;
     int *count_return;
     Atom property;
```

The XGetRGBColormaps routine can generate BadAtom and BadWindow errors.

The XGetRGBColormaps routine supersedes XGetStandardColormap.

The XSetRGBColormaps routine replaces (or sets if the property does not exist) the RGB colormap definition in the indicated property on the specified window. The property is stored with a type of RGB_COLOR_MAP and a format of 32. It is the caller's responsibility to honor the ICCCM's restriction that only RGB_DEFAULT_MAP contain more than one definition.

```
void XSetRGBColormaps (dpy, w, stdcmap, count, property)
    Display *dpy;
    Window w;
    XStandardColormap *stdcmap;
    int count;
    Atom property;
```

The XSetRGBColormaps routine can generate BadAlloc, BadAtom, and BadWindow errors.

The XSetRGBColormaps routine supersedes XSetStandardColormap.

C.2.1.2.9 Convenience Routines – In ULTRIX/UWS Version 2.2, the following routines were added to provide convenient interfaces to new property formats:

The XSetWMProperties routine provides a convenient interface for setting the essential properties on the specified window for communicating with other clients (particularly window and session managers).

```
void XSetWMProperties (dpy, w, window_name, icon_name, argv, argc,
                       normal_hints, wm_hints)
      Display *dpy;                 /* user's display connection */
      Window w;                     /* window to decorate */
      XTextProperty *window_name;   /* name of application */
      XTextProperty *icon_name;     /* name string for icon */
      char **argv;                  /* command line */
      int argc;                     /* size of command line */
      XSize_hints *normal_hints;    /* size hints in normal state */
      XWM_hints *wm_hints;          /* misc. window manager hints */
```

If window_name is nonnull, the WM_NAME property is set using XSetWMName. If icon_name is nonnull, WM_ICON_NAME is set using XSetWMIconName. If argv is nonnull, WM_COMMAND is set using XSetCommand (note that an argc of 0 is allowed to indicate a zero-length command). The hostname of this machine is stored using XSetWMClientMachine. If normal_hints is nonnull, WM_NORMAL_HINTS is set using XSetWMNormalHints. If wm_hints is nonnull, WM_HINTS is set using XSetWMHints.

The XSetWMProperties routine can return any of errors generated by the routines mentioned above.

The XSetWMProperties routine supersedes XSetStandardProperties.

The XWMGeometry routine combines geometry information (given in the format used by XParseGeometry) specified by you and by the calling program with size hints (usually the ones to be stored in WM_NORMAL_HINTS) and returns the location, size, and gravity (NorthWestGravity, NorthEastGravity, SouthEastGravity or SouthWestGravity) describing the window. If the base size is not set in the hints structure, then the minimum size will be used if set; otherwise, a base size of 0 is assumed. If no minimum size is set in the hints structure, the base size is used.

```
int XWMGeometry (dpy, screen, user_geom, def_geom, bwidth, hints,
            x_return, y_return, width_return, height_return,
             gravity_return)
  Display *dpy;                       /* user's display connection */
  int screen;                         /* screen queried */
  char *user_geom;                    /* user provided geometry spec */
  char *def_geom;                     /* default window geometry spec */
  unsigned int bwidth;                /* border width */
  XSizeHints *hints;                  /* usually WM_NORMAL_HINTS */
  int *x_return, *y_return;           /* set location if successful */
  int *width_return, *height_return; /* set size if successful */
  int *gravity_return;                /* coordinates' gravity */
```

Invalid geometry specifications may cause a width or height of 0 to be returned. The caller may pass the address of the hints win_gravity field as gravity_return to update the hints directly.

The `XWMGeometry` routine supersedes `XGeometry`.

C.2.1.3 Obsolete Functions in Xlib – The following functions in Xlib became obsolete starting in ULTRIX/UWS Version 2.2:

`XGeometry`	Superseded by `WMGeometry`
`XGetNormalHints`	Superseded by `GetWMNormalHints`
`XGetSizeHints`	Superseded by `GetWMSizeHints`
`XGetStandardColormap`	Superseded by `GetRGBColormaps`
`XGetZoomHints`	No longer supported by ICCCM
`XSetNormalHints`	Superseded by `SetWMNormalHints`
`XSetSizeHints`	Superseded by `SetWMSizeHints`
`XSetStandardColormap`	Superseded by `SetRGBColormaps`
`XSetStandardProperties`	Superseded by `SetWMProperties`
`XSetZoomHints`	No longer supported by ICCCM

C.2.1.4 Obsolete Constants in Xutil.h – The following constants became obsolete in `Xutil.h` starting in ULTRIX/UWS Version 2.2:

`DontCareState`	No longer supported by ICCCM
`InactiveState`	No longer supported by ICCCM
`ZoomState`	No longer supported by ICCCM

C.2.1.5 Convenience Functions – The following functions were marked in ULTRIX/UWS Version 2.2 as convenience functions for window and icon names encoded as STRING properties:

`XFetchName`	Generalized by `XGetWMName`
`XGetIconName`	Generalized by `XGetWMIconName`
`XSetIconName`	Generalized by `XSetWMIconName`
`XStoreName`	Generalized by `XSetWMName`

C.2.1.6 Changes to Xlib Interfaces – In ULTRIX/UWS Version 2.2, the following changes were made to the Xlib interfaces that correct omissions in previous releases.

C.2.1.6.1 Getting Screen Number from Screen Pointer – A member containing the number of the screen has been added to the opaque `Screen` structure and a macro and function have been added to access the index. This allows applications that use screen pointers instead of indices with routines that take screen numbers. Xlib tends to use screen numbers rather than screen pointers.

```
int ScreenNumberOfScreen(scr)
int XScreenNumberOfScreen(scr)
    Screen *scr;
```

The `ScreenNumberOfScreen` macro and `XScreenNumberOfScreen` functio n return the screen index corresponding to the indicated screen pointer.

C.2.1.6.2 Pixmap Formats – To provide an interface to the Pixmap Format information returned in the connection setup block, the following structure analogous to `XGCValues` has been added to `Xlib.h`:

```
typedef struct {
    int depth;
    int bits_per_pixel;
    int scanline_pad;
} XPixmapFormatValues;
```

In addition, the following function for retrieving the information has also been added:

```
XPixmapFormatValues *XListPixmapFormats (dpy, count)
    Display *dpy;
    int *count;
```

The `XListPixmapFormats` function returns an array of `XPixmapFormatValues` structures describing the types of Z format images supported by the display. The argument count is set to the number of pixmap formats supported by the display. The storage for the returned structures can be released by calling `XFree`. If insufficient memory is available, NULL is returned.

C.2.1.6.3 Returning Old Error Handlers – To make nesting of error handlers possible, the declarations of the routines `XSetErrorHandler` and `XSetIOErrorHandler` return the previous handler, are as follows:

```
int (*XSetErrorHandler (handler))()
int (*XSetIOErrorHandler (handler))()
int (*handler)();
```

C.2.1.6.4 Getting XGCValues from a GC – To make it easier to reuse GCs, a new routine was added in the ULTRIX/UWS Version 2.2 release for retrieving the contents of a GC that the `XGCValues` structure can set.

```
Status XGetGCValues (dpy, gc, valuemask, values_return)
    Display *dpy;
    GC gc;
    unsigned long valuemask;
    XGCValues *values_return;
```

The `XGetGCValues` function returns the fields indicated by valuemask of the specified graphics context. The clip mask and dash list (represented by the GCClipMask and GCDashList bits in valuemask) can not be requested. If valuemask contains a valid set of GC mask bits and no errors occur, the requested fields in `values_return` are set and a nonzero status is returned; otherwise, a status of 0 is returned. Valid GC mask bits are:

```
GCArcMode           GCBackground
GCCapStyle          GCClipXOrigin
GCClipYOrigin       GCDashOffset
GCFillRule          GCFillStyle
GCFont              GCForeground
GCFunction          GCGraphicsExposures
GCJoinStyle         GCLineStyle
GCLineWidth         GCPlaneMask
GCStipple           GCSubwindowMode
GCTile              GCTileStipXOrigin
GCTileStipYOrigin
```

C.2.2 DECwindows Toolkit Programming

This section describes new and changed features pertaining to the DECwindows programming toolkit since ULTRIX/UWS Version 2.1:

C.2.2.1 New Widgets and Gadgets – This section describes the widgets and gadgets that have been added to the Toolkit since ULTRIX/UWS Version 2.1.

- Pulldown Menu Entry Gadgets
- Color Mix Widget
- Compound String Text Widget

C.2.2.1.1 Pulldown Menu Entry Gadgets – In ULTRIX/UWS Version 4.0, the DECwindows Toolkit provides pull-down menu entry gadgets and widgets. As with gadget variants of other widgets, they should be used unless the gadget does not provide enough flexibility.

The `DwtPullDownMenuEntryCreate` routine has been added to the Toolkit. This routine creates a pull-down menu entry gadget. See the *Guide to the XUI Toolkit: C Language Binding* manual for a description of this routine.

C.2.2.1.2 Color Mix Widget – In ULTRIX/UWS Version 2.2, the color mixing widget was added. This widget enables end users to define colors. It is a pop-up dialog box with two components: a color display region and a color mixing tool. The default color display region contains two color tiles: one presents the original color, and the other shows the new color as it is modified by the user. The default mixing tool follows the RGB colormodel.

Both the color display region and mixing tool can be replaced with custom components by the application. The color values returned to the application are standard X Window System, Version 11, RGB values (0 through 65,535). Note that the color mixing widget is simply a mechanism for applications to query users for a color. It does not allocate resources (color cells) for the application.

The color mix widget includes the following routines:

- `DwtColorMixCreate` - Creates the color mix widget (low-level create only).
- `DwtColorMixGetNewColor` - Gets the current RGB values of new color.
- `DwtColorMixSetNewColor` - Sets the current RGB values of new color.

C.2.2.1.2 Hue Lightness Saturation (HLS) Colormodel – In ULTRIX/UWS Version 4.0, the Hue Lightness Saturation colormodel was added. The XUI color mix widget now supports both the HLS and RGB colormodels. This functionality is built into the color mixer. No code changes are required and the same X11 RGB callback information is returned regardless of which colormodel is used. The impact is only to end users. A new option menu presents the colormodel choices HLS and RGB giving users the ability to switch between models at will.

There are 11 new resources (10 of which are labels):

- `DwtNcolorModel` colormodel currently being used. Choices are `DwtColorModelHLS` (the default), and `DwtColorModelRGB`.

Note

It is assumed that applications probably will only set this resource before the widget is managed (if at all), and will allow users to switch colormodels themselves using the option menu.

- `DwtNhueLabel` label of hue scale widget. The default is "Hue:".
- `DwtNlightLabel` label of lightness scale widget. The default is "Lightness:".
- `DwtNsatLabel` label of saturation scale widget. The default is "Saturation:".
- `DwtNblackLabel` label for zero end of lightness scale widget. The default is "Black".
- `DwtNwhiteLabel` label for 100% end of lightness scale widget. The default is "White".
- `DwtNgrayLabel` label for zero end of saturation scale widget. The default is "Gray".
- `DwtNfullLabel` label for 100% end of saturation scale widget. The default is "Full".
- `DwtNoptionLabel` label for colormodel option menu. The default is "Color Model: ".
- `DwtNhlsLabel` label for colormodel option menu HLS option. The default is "HLS".
- `DwtNrgbLabel` label for colormodel option menu RGB option. The default is "RGB".

C.2.2.1.3 Colormix Red, Green and Blue Labels – There is a problem in the toolkit color mixing widget such that the Red, Green, and Blue labels cannot be modified after widget creation. You must specify label changes during widget creation.

C.2.2.1.4 Attached Dialog Box Widget – The attached Dialog Box widget does not retain offsets after widget creation. The values returned by `XtGetValues` are zero, instead of what they were when the widgets were created.

C.2.2.1.5 Compound String Text Widget – The compound string text widget supports the same routines and attributes as `SText`, except that it uses compound strings instead of C null-terminated strings.

The compound string text widget includes the following routines:

- CS TEXT - Create cs text widget (high-level call).
- CS TEXT CREATE - Create cs text widget (low-level call).
- CS TEXT CLEAR SELECTION - Clears global selection.

- CS TEXT GET EDITABLE - Obtains current permission state.
- CS TEXT GET MAX LENGTH - Obtains maximum allowable text length.
- CS TEXT GET SELECTION - Retrieves global selection.
- CS TEXT GET STRING - Retrieves current text.
- CS TEXT REPLACE - Replaces a portion of the current text.
- CS TEXT SET EDITABLE - Sets permission state.
- CS TEXT SET MAX LENGTH - Sets maximum allowable text length.
- CS TEXT SET SELECTION - Makes specified text current global selection.
- CS TEXT SET STRING - Sets text.

C.2.2.2 New Routines – This section describes additions to the following routines that have been added since ULTRIX/UWS Version 2.1:

- Low Level Routines
- Compound String Routines
- Cut and Paste Routines
- Convenience Routines

C.2.2.2.1 New Resources for Low-Level Toolkit Routines – The XUI Toolkit provided new resources for low-level routines in the ULTRIX/UWS Version 2.2 release. The following table lists some of those new widget attributes and corresponding resource names, and provides descriptions of the attributes and their associated default values.

FILE SELECTION WIDGET	
Attribute:	`file_to_extern_proc`
Resource Name:	`DwtNfileToExternProc`
Description:	Converts native internal file names to custom external file names displayed to the user.
Default Value:	NULL
Attribute:	`file_to_intern_proc`
Resource Name:	`DwtNfileToInternProc`
Description:	Converts custom external file names, displayed to the user, to native internal file names.
Default Value:	NULL
Attribute:	`mask_to_extern_proc`
Resource Name:	`DwtNmaskToExternProc`
Description:	Converts native internal directory masks to custom directory masks displayed to the user.
Default Value:	NULL
Attribute:	`mask_to_intern_proc`
Resource Name:	`DwtNmaskToInternProc`
Description:	Converts custom external directory masks,

FILE SELECTION WIDGET

	displayed to the user, to internal directory masks.
Default Value:	NULL

HELP WIDGET

Attribute:	`gototopic_label`
Resource Name:	`DwtNgototopicLabel`
Description:	The label for the Go To Topic menu entry.
Default Value:	"Go To Topic"

Attribute:	`gobacktopic_label`
Resource Name:	`DwtNgobacktopicLabel`
Description:	The label for the Go Back push button in the help widget window.
Default Value:	"Go Back"

Attribute:	`visittopic_label`
Resource Name:	`DwtNvisittopicLabel`
Description:	Label for the Visit Topic menu item in the View pull-down menu.
Default Value:	"Visit Topic"

Attribute:	`close_label`
Resource Name:	`DwtNcloseLabel`
Description:	Label for the Exit push button in the help widget window.
Default Value:	"Exit"

Attribute:	`helphelp`
Resource Name:	`DwtNhelphelpLabel`
Description:	The label for the Overview menu item in the Using Help pull-down menu.
Default Value:	"Overview"

Attribute:	`helpontitle_label`
Resource Name:	`DwtNhelpontitleLabel`
Description:	Label for the help widget title bar used in conjunction with the applications name.
Default Value:	"Help On"

Attribute:	`help_acknowledge_label`
Resource Name:	`DwtNhelpAcknowledgeLabel`
Description:	Label for the Acknowledge push button in the error message box.
Default Value:	"Acknowledge"

Attribute:	`help_on_help_title`
Resource Name:	`DwtNhelpOnHelpTitle`
Description:	Label for the title bar in the Help-on-Help widget.
Default Value:	"Using Help"

HELP WIDGET

Attribute:	`cache_help_library`
Resource Name:	`DwtNcacheHelpLibrary`
Description:	A Boolean attribute that specifies whether or not the text of the help library is stored in the help widget's cache memory.
Default Value:	FALSE

Attribute:	`map_callback`
Resource Name:	`DwtNmapCallback`
Description:	The callback routine or routines called when the help widget is mapped.
Default Value:	NULL

MENU WIDGET

Attribute:	`change_vis_ atts`
Resource Name:	`DwtNchangeVisAtts`
Description:	A Boolean specifying whether a menu can modify the visual attributes of its children. When this attribute is FALSE, a menu widget cannot make changes to its children.
Default Value:	TRUE

Attribute:	`menu_extend_last_row`
Resource Name:	`DwtNmenuExtendLastRow`
Description:	The width of the menu (for vertical menus) or the height of the menu (for horizontal menus).
Default Value:	TRUE

POP-UP DIALOG WIDGET

Attribute:	`auto_unrealize`
Resource Name:	`DwtNautoUnrealize`
Description:	A Boolean that specifies whether a dialog box unrealizes itself when it is unmanaged.
Default Value:	FALSE

MESSAGE BOX WIDGET

Attribute:	`second_label`
Resource Name:	`DwtNsecondLabel`
Description:	The text for a secondary label.
Default Value:	NULL

Attribute:	`label_alignment`
Description:	The alignment of the primary label.
Default Value:	`AlignmentCenter`

MESSAGE BOX WIDGET

	`AlignmentBegin` `AlignmentEnd`
Attribute:	`second_label_alignment`
Resource Name:	`DwtNsecondLabelAlignment`
Description:	The alignment of the secondary label.
Default Value:	`DwtAlignmentBeginning`
Attribute:	`icon_pixmap`
Resource Name:	`DwtNiconPixmap`
Description:	The pixmap used for the icon.
Default Value:	"!" for caution box icon "*" for message box icon wait cursor (watch) for work box icon

PUSH BUTTON WIDGET

Attribute:	`insensitive_pixmap`
Resource Name:	`DwtNinsensitivePixmap`
Description:	The pixmap used when the push button is set to insensitive. Applies only if push button label is specified as pixmap.
Default Value:	NULL

SCROLL BAR CREATE WIDGET

Attribute:	`show_arrows`
Resource Name:	`DwtNshowArrows`
Description:	A Boolean indicating whether or not scroll bar has stepping arrows.
Default Value:	TRUE

STEXT WIDGET

Attribute:	`user_data`
Resource Name:	`DwtNuserData`
Description:	Any private data to be associated with the textwidget. The XUI Toolkit does not interpret this data.
Default Value:	NULL

TOGGLE BUTTON WIDGET

Attribute:	`insensitive_pixmap_on`
Resource Name:	`DwtNinsensitivePixmap`
Description:	Displayed when the button state is TRUE

TOGGLE BUTTON WIDGET	
	and widget is insensitive.
Default Value:	NULL
Attribute:	`insensitive_pixmap_off`
Resource Name:	`DwtNinsensitivePixmapOff`
Description:	Displayed when button state is FALSE and widget is insensitive.
Default Value:	NULL

C.2.2.2.2 New Compound String Routines – The following new compound string routines have been added since the ULTRIX/UWS Version 2.1 release:

- STRING INIT CONTEXT -- Initializes the context required by GET NEXT SEGMENT
- STRING FREE CONTEXT -- Frees a compound string context structure

For more information see the section Performance of Init Get Segment in these release notes.

C.2.2.2.3 Cut and Paste Routines – The following Cut and Paste routines have been added since ULTRIX/UWS Version 2.1:

- Start Copy to Clipboard -- Identical to Begin Copy to Clipboard, except that the time stamp of the event is included.
- Start Copy from Clipboard -- Indicates that the application is ready to start copying data from the clipboard.
- End Copy from Clipboard -- Indicates that the application has completed copying data from the clipboard.
- Clipboard Register Format -- Registers the length of the data for formats not specified by the *Inter-Client Communications Conventions Manual* (ICCCM).

C.2.2.2.4 New Convenience Routines – The following convenience routines have been added since ULTRIX/UWS Version 2.1:

- ACTIVATE WIDGET CONVENIENCE ROUTINE -- Provides a mechanism for applications to activate a UWS Toolkit pushbutton. This is useful in cases where buttons present actions also available in menus. For example, in DECwindows mail you can reply to a message either by activating the REPLY button or by selecting the Reply menu item. If the menu item is selected, ACTIVATE WIDGET can be used to flash the REPLY button presenting a more consistent UI.
- GET USER DATA CONVENIENCE ROUTINE -- Provides a short-cut for retrieving the widget user data field. Returns the user data associated with the widget.

C.2.2.3 Bug Fixes and Other Changes – This subsection describes problem fixes and miscellaneous changes to the XUI Toolkit since the ULTRIX/UWS Version 2.1 release.

C.2.2.3.1 Changes to Existing Convenience Routines – The UWS Version of the XUI Toolkit corrected the binding for number children `DwtNumChildren`.

C.2.2.3.2 DEC Windows Resource Manager (DRM) – Both the Intrinsics and the DECwindows resource manager (DRM) can now be initialized as many times as required by the application. This is an extension of the MIT R3 Intrinsics and should not be used by applications that want to remain R3 compatible.

C.2.2.3.3 Internal Format of Compound Strings – Beginning with the ULTRIX/UWS Version 2.2 release, Compound strings are stored in CDA format. This change is transparent to applications that treat compound strings as opaque entities.

C.2.2.3.4 Performance of INIT GET Segment – In the ULTRIX/UWS Version 2.2 release, the change to compound strings significantly decreased the performance of the `DwtInitGetSegment` when used to fetch multiple segments from a compound string. Because of this, the STRING INIT CONTEXT and STRING FREE CONTEXT routines were added and should be used for better performance.

C.2.2.3.5 dwtappli.h – Two erroneous declarations are contained in `dwtappli.h` They are:

- `DwtGetSlider`
- `DwtSetSlider`

C.2.2.3.6 Font Units – In this release, the XUI Toolkit uses the AVERAGE_WIDTH and RESOLUTION_Y properties.

C.2.2.3.7 Destroy Callback – Unlike other toolkit callbacks, the destroy callback returns only two arguments: widget id and tag. The reason argument is NULL. Applications therefore should avoid setting destroy callbacks to call general callback routines (handling numerous actions such as activate, arm, disarm, and so forth) that depend on a reason argument.

C.2.2.3.8 Listbox Dynamic Sizing – The proper way to change `listbox` width is through the `SetValues` attribute. `listbox` does not support dynamic dimension changes. Therefore, placing a `listbox` inside an attached dialog box, with attachments to both the left and right side of the attached dialog box, may lead to the items selectable area not spanning the full width of the `listbox`. This is because when attached dialog boxes change size they dynamically resize their children.

Also note that the proper way to change the `listbox` height is through the `ItemsCount` attribute. Modifying the height attribute will not change the number of visible items. For example, doubling the `listbox` height but not modifying the `ItemsCount` attribute results in a `listbox` only half full of items, with the remaining area left blank.

There is a problem in `listbox` such that if resize is Fixed (`FALSE`) and a wide item is added in the visible region of the `listbox` the List Box menu does not grow and the horizontal scroll bar fails to get updated. This behavior only occurs if the added item is visible. A workaround is always to add the item in the nonvisible region of `listbox`. For example:

```
(1) DwtListBoxSetPos (top)

(2) DwtListBoxAdditem (do not specify a position, it will be added at
                       the bottom of the list - non-visible area)
```

C.2.2.3.9 Help Widget Listbox – Under certain circumstances, the help widget's listbox selectable area does not span the entire width of the widget. However, items may still be selected by clicking the mouse button on the item text.

C.2.2.3.10 DECwindows Toolkit and the MIT R3 Intrinsics. – The version number of the Intrinsics supplied as part of the DECwindows kit does not match the MIT R3 Intrinsics. The MIT R3 Intrinsics version number is 11003, while the DECwindows Intrinsics are 7001.

The DECwindows `XtNameToWidget` routine does not conform to the MIT R3 Intrinsics. The specification states that the first component of the names parameter is matched against the children of the passed reference widget; the implementation matches the first component of the names parameter against the reference widget, not the children. Thus to use the DECwindows version, add the name of the reference widget to the beginning of the name list.

C.2.2.3.11 Selection Pushbuttons – Setting the OK and CANCEL pushbutton labels to NULL or to empty strings does not remove the pushbuttons as stated in the documentation; instead, it results in blank labels.

C.2.2.3.12 Using Accelerators on Pushbutton and Togglebutton Gadgets – Only the first gadget child of a widget parent may have a "#" operator, such as #override, in its button accelerator specification. All gadget button accelerators of a widget parent will have the same # operator as the first gadget child.

C.2.2.3.13 Generating Widget/Gadget Exposes – Generating widget/gadget exposes by calling `SetValues` without visual changes is not supported.

In previous versions of the XUI toolkit, some widgets incorrectly redisplayed after `SetValues` whenever the arglist contained a visual field, even if that field did not change. For example, an application could initiate a pushbutton redraw by passing an unchanged borderwidth in `SetValues` if the widget was a child of a dialog box using font units.

Applications can redraw widgets either by changing a visual field or by calling `XClearArea` on the widget window.

C.2.2.3.14 Toggle Button Set State Routine – A problem in the toolkit `DwtToggleButtonSetState` affects toggle buttons with on/off pixmaps. If the widget has not been realized, `SetState` correctly updates the toggle button value but not the on/off pixmap. When realized, they display the wrong pixmap.

A simple workaround is to use the `SetValues` mechanism instead setting `DwtNvalue` to TRUE. This correctly updates the pixmap as well as the toggle button value regardless of whether the widget is realized.

C.2.2.3.15 Toggle Button Gadgets – Toggle button gadgets no longer redraw themselves after applications change their value through `SetValues` (although their values do change). This problem only affected visible toggle button gadgets modified through `SetValues`.

A problem introduced in the MIT R3 intrinsics that prevented applications that read in user's X defaults files from opening more than one display has been fixed.

C.2.2.3.16 Dialog Box Race Condition – XUI Toolkit dialog boxes perform an `XGrabKey` on the TAB key so that they can synchronously transfer focus to the next child within the Dialog Box. If a Dialog Box receives a TAB key while the Toolkit is filtering events (for example, while another modal dialog box is up), the original Dialog Box does not see the TAB event and never calls `XAllowEvents` to unfreeze the keyboard. If this happens, you must exit the application and restart it to unfreeze the keyboard.

C.2.2.3.17 Right to Left Compound Strings – Right to left compound strings are displayed left to right in dialog box title resource.

C.2.2.3.18 DwtResolvePartOffsets Function – This function is obsolete. The -DNOT_VMS_V1 flag has been removed. Any programs that depend on this function must be aware that the flag is no longer valid. All interfaces (both VAX and RISC) should act as if the flag were set at all times.

C.2.2.3.19 Delete Sub-Menu – If an application destroys the original sub-menu and immediately updates the field with the new menu using `SetValues`, the sub-menu field is updated with the new widget id. However, because destroy is a two-phase process, the menu does not know its original sub-menu has been destroyed and when the second phase executes (later in the Mainloop), the parent menu is informed that its sub-menu has been destroyed and sets that field to NULL even though it now points to the new widget. This can result in an "X - NOT A VALID Window" error.

The workaround is to return to `Mainloop` and wait a sufficient period of time to allow the second phase of destroy to complete (and menu to clear its sub-menu field) before updating using `SetValues`.

C.2.2.3.20 Size of Core – Do not count on the size of the core part record as a fixed size; obtain the size at widget initialization time. This will prevent you from having to recompile programs whenever a new version of UWS is released. Indices and part offsets are used to access fields in your own widgets. For further information, see the *Guide to the XUI Toolkit: C Language Binding*.

C.2.2.3.21 DwtWidget.h File – The following problem in ULTRIX/UWS Version 2.2 was fixed in ULTRIX/UWS Version 4.0:

When compiling a program that includes `DwtWidget.h`, or `DECDwtWidgetProg.h` that is linked to `DwtWidget.h`, fatal syntax errors were reported due to the typedef `Object` not being declared.

C.2.2.3.22 Option Menus – Option menus are documented as being able to have a parent that is a shell widget. However, only a menu shell widget can be the parent of an option menu widget. If you attempt to parent an option menu widget with any other type of shell widget an empty, transparent window will be displayed with no menu button available.

To create an application that only uses one option menu make the option menu a child of a popup menu. The following code fragment provides an example of this:

```
include <X11/DwtAppli.h>
     .
     .
     .
Widget toplevel, popup, options;
Widget buttons[3];
int num_args;
Arg args[1];

toplevel = XtInitialize("example", "Example",
               (XrmOptionDescRec *) NULL,
               0, &argc, argv);

popup = DwtMenuPopupCreate(toplevel, "Popup1", (ArgList) NULL, 0);

buttons[0] = DwtPushButtonGadgetCreate(popup, "a", (ArgList) NULL, 0);
buttons[1] = DwtPushButtonGadgetCreate(popup, "b", (ArgList) NULL, 0);
buttons[2] = DwtPushButtonGadgetCreate(popup, "c", (ArgList) NULL, 0);
XtManageChildren(buttons, 3);

num_args = 0;
XtSetArg(args[num_args], DwtNsubMenuId, popup);
num_args++;
options = DwtOptionMenuCreate(popup, "Options", args, num_args);
XtManageChild(options);

XtRealizeWidget(toplevel);
     .
     .
     .
```

C.2.2.3.23 Popup Dialog Boxes – A problem in the intrinsics allows popup dialog boxes with no icon button, `DwtNnoIconify` set TRUE, to be initially created iconified `DwtNiconic` TRUE. The "iconified" popup does not have an icon box and cannot be popped up. Additionally, attempting operations such as `SetInputFocus` on the popup will lead to an access violation.

C.2.2.3.24 New and Omitted Widget Arguments – Several widget arguments have been added since ULTRIX/UWS Version 2.2:

```
attached_dialog_box:    direction_r_to_l
caution_box:            direction_r_to_l
color_mix:              hue_label, light_label, sat_label, black_label,
                        white_label,  gray_label, full_label, option_label,
```

```
command_window:         direction_r_to_l
dialog_box:             direction_r_to_l
menu_bar:               menu_extend_last_row, direction_r_to_l
popup_menu:             menu_extend_last_row, direction_r_to_l
pulldown_menu:          menu_extend_last_row, direction_r_to_l
radio_box:              menu_extend_last_row, direction_r_to_l
work_area_menu:         menu_extend_last_row, direction_r_to_l
list_box:               spacing, direction_r_to_l
file_selection:         auto_unmanage, auto_unrealize, direction_r_to_l
selection:              auto_unmanage, auto_unrealize, direction_r_to_l
help_box:               direction_r_to_l
main_window:            direction_r_to_l
message_box:            direction_r_to_l
option_menu:            direction_r_to_l
popup_attached_db:      direction_r_to_l
popup_dialog_box:       direction_r_to_l
scale:                  direction_r_to_l
scroll_bar:             direction_r_to_l
scroll_window:          direction_r_to_l
separator:              direction_r_to_l
window:                 direction_r_to_l
work_in_progress_box:   direction_r_to_l
```

C.2.2.3.25 Constraint Attributes – Constraint attributes are no longer allowed on nonconstraint widgets.

C.2.2.3.26 Large Value Tables – Large value tables now fit into DRM context buffers.

C.2.2.3.27 `DEC_KANJI` or `DEC_HANZI` as the Default Character Set – Setting DEC_KANJI or DEC_HANZI as the default character set no longer incorrectly generates the error "support for this character set may be removed in a future release."

C.2.2.3.28 Large Pixmaps – Pixmaps that are larger than the size of offscreen memory cause the server to send an error and cause the client to crash.

To avoid this problem, always allocate pixmaps that are no larger than the available offscreen memory.

C.2.2.4 User Interface Language (UIL) – The following is a list of the new UIL features since ULTRIX/UWS Version 2.1:

- Continuous image support (color and gray scale)
- Multiple callback procedures per reason
- Direct support of constraint arguments
- Integer tables (for use as tag values in callbacks)
- Support for CS Text widget
- Support for `XBITMAPFILE` function (reads X bitmap files from disk)
- Support for new compound string routines
- Support for multiple segment compound string creation using either the specified or the default character set

The following arguments are not directly available in the UIL compiler for ULTRIX/UWS Version 4.1:

Argument	Data Type	Default Value	Valid for These Objects
auto_unmanage	Boolean	True	selection file_selection
auto_unrealize	Boolean	False	selection file_selection
direction_r_to_l	integer	DwtDirectionRightDown	color_mix
grab_key_syms	translation_table	Default translation table syntax	color_mix
menu_extend_last_row	Boolean	True	menu_bar popup_menu pulldown menu radio_box work_area_menu
no_resize	Boolean	True	color_mix
take_focus	Boolean	True (modal); False (modeless)	color_mix

To set one of these arguments in a UIL module, use the ARGUMENT function to define the argument. See the *Guide to the XUI User Interface Language Compiler* for information on the ARGUMENT function.

C.2.3 DECwindows Applications Changes

This section discusses changes to DECwindows applications in ULTRIX/UWS Version 2.2 and ULTRIX/UWS Version 4.0.

C.2.3.1 CDA Viewer - dxvdoc – The menus for the `dxvdoc` program have changed slightly to conform better with other UWS applications. The Set-Up... menu item has moved from the File menu to the Customize menu and has been renamed Options.... It provides the same functionality as the old Set-Up... menu item, allowing you to change processing options.

The `dxvdoc` program now starts up in the center rather than the upper left of the screen.

C.2.3.2 Calculator - dxcalc – All text in `dxcalc` is now in UIL files and the following features have been added since ULTRIX/UWS Version 2.1:

- Context-sensitive help for each key
- Support for standard Edit menu accelerators
- A Customize Menu with standard, savable features

C.2.3.3 Cardfiler - dxcardfiler – All text in `dxcardfiler` is now in UIL files and the following features have been added since ULTRIX/UWS Version 2.1:

- Context-sensitive help for each key
- Support for standard menu accelerators
- A Customize Menu with standard, savable features

C.2.3.4 Clock - dxclock – All text in `dxclock` is now in UIL files and the following features have been added since ULTRIX/UWS Version 2.1:

- Context-sensitive help for each key
- A Customize Menu with standard, savable features
- A display that uses less screen space. The new arrangement allows for use of a smaller window while keeping all the information displayed and readable.
- A 24-hour time display

C.2.3.5 Notepad - dxnotepad – Since ULTRIX/UWS Version 2.2, the `dxnotepad` application has the following new features:

- Edit
- Search
- Navigate
- Customize
- In addition, please note the following change to the UNDO command:

  Selecting all the text in `dxnotepad`, indenting plus or minus four spaces and selecting UNDO causes all the text to disappear until you select UNDO a second time or select REDO once. Thus, it appears that UNDO loses all your text when it really does not.

C.2.3.6 PostScript Previewer - dxpsview – The following are changes to `dxpsview` since ULTRIX/UWS Version 2.1:

- Skipping from page to page is faster on structured files. This works only if the files are structured and commented correctly. If you are unable to preview a file, try toggling the Use Comments option under the Options menu.
- You now can specify whether `dxpsview` draws images directly to the screen or stores images until an entire page is ready to display using the Watch Progress option from the Options menu.
- You can change the scale by which images are magnified or shrunk. To change the scale, chose the Sheet Selection option from the Option menu. See the "Scale Factors Larger than 2.0" note in Chapter 4 of these release notes for an important caveat regarding scale factor usage.

C.2.3.7 Puzzle - dxpuzzle – All text in dxpuzzle is now in UIL files and the following features have been added since ULTRIX/UWS Version 2.1:

- Context-sensitive help is available for each key
- A Customize Menu with standard, savable features

C.2.3.8 Session Manager - dxsession – The following notes describe changes to the Session Manager since ULTRIX/UWS Version 2.1:

C.2.3.8.1 Customize Language – The Customize Language dialog box now allows you to specify a language (for example, English US, French) for subsequent applications. The language list also includes a selected entitled Default that indicates a default set of user interface specifications that are installation dependent. These are used if language-specific specifications are unavailable.

If you select a language, the `dxsession` program sets the `xnlLanguage` resource for all applications, and records this setting in the user's `.Xdefaults` file. It also sets the `$LANG` environment variable. If you use the `Default` option, the `xnlLanguage` resource is removed from the root property and the user's `.Xdefaults` file, and the `$LANG` environment variable is set to the empty string.

Note that the `dxsession`'s use of the `$LANG` variable is inconsistent with the naming conventions used by NLS (Native Language System). This inconsistent use of the `$LANG` variable may cause XPG3 conforming internationalized applications to behave in an unexpected fashion. For example, applications will not be able to change their locale based on the `dxsession`'s setting of `$LANG`. See the `lang(5int)` reference pages for a description of supported languages.

Note

For a language to take effect, it must be installed and licensed.

C.2.3.8.2 Customize Window – You can now change the window manager through the Customize Window dialog box that will change the resource value for `sm.windowManagerName` in your `.Xdefaults` file.

If you decide to use the default window manager, the session manager will use the value of the `sm.windowManagerName_default` resource. Typically, this is set only in the app-defaults file for the session manager. If no such value exists, the default window manager value default is `/usr/bin/dxwm`.

C.2.3.8.3 New Per-View Resources – There are some new per-view resources; they are `iconFormat` and `titleFormat.` The value of these are `printf`-like format strings with the following conventions:

```
%v is replaced by the view name
%d is replaced by the current directory
```

To specify a field width, replace % with *%n*, where *n* is the field width to be used. If you specify a negative number, the field will be left-justified. Otherwise, the field will be right-justified, which is the default. To specify a literal newline, use `\n`. To specify a literal tab, use `\t`.

The `iconMaxDirSegments` and `titleMaxDirSegments` resources provide further control over the contents of icons and title bars. They specify how many

segments of the current directory appear in the place of `%d` in a format string. If `iconMaxDirSegments` is 2, then the last two segments of the path will appear in the icon.

Views can now be customized individually by name. The resource name of the background color of the startup view is `*Startup.background`. The resource names of the icon format and title bar format resources are also at this level. The names of these resources of the startup view are:

```
*Startup.iconFormat
*Startup.titleFormat
*Startup.iconMaxDirSegments
*Startup.titleMaxDirSegments
```

New views that are not named have the name `Unnamed`. Changing the name of a view does not cause new resources to be applied to it. A renamed view retains the resources of its original name.

C.2.3.9 User Executive - dxue – Since ULTRIX/UWS Version 2.1, the `dxue` application has changed some resources and some of its widget hierarchy. Its class name has changed to `Executive` and its instance name is `argv[0]`, which is the name by which it is invoked. Any resources supplied in your `.Xdefaults` file that are specific to `dxue` must change to use this new name.

C.2.4 Font Format Changes

When you upgrade from ULTRIX/UWS Version 4.0 to ULTRIX/UWS Version 4.1, no changes to screen fonts are required. Any layered application that runs under ULTRIX/UWS Version 2.1 and bundles screen fonts might be affected by changes in the ULTRIX/UWS Version 4.0 and ULTRIX/UWS Version 4.1 font formats and file extensions. If the application bundled the BDF form of the fonts, you can upgrade them, using the `dxfc` command (or the `dxfc3d` command for a `Xgb` server). However, if the application bundled only the compiled form of the fonts, you cannot automatically upgrade the fonts to use with ULTRIX/UWS Version 4.0 or ULTRIX/UWS Version 4.1

C.2.5 Conformance to Standards

While Xlib is ICCCM compliant, the Intrinsics are not. Clients that subclass the Shell widget must recompile in order to work with Xlib Release 4.

RISC-VAX System Differences and Porting Hints D

This appendix discusses the following topics:

- The differences between RISC-based and VAX-based systems
- Some hints for porting software to RISC-based systems

D.1 Differences Between RISC-Based and VAX-Based Systems

The following components differ on RISC-based and VAX-based systems:

- Executable images

  Executable images on RISC-based systems are larger and therefore take up more disk space than their counterparts on VAX-based systems. This is due to the instruction set of the RISC architecture. Typically, images on RISC-based systems are 30 to 40 percent larger than on VAX-based systems.

- Shared Memory Segments (SMS)

  The attach points for Shared Memory Segments (SMS) in the virtual address space of a process on RISC-based systems are different than on VAX-based systems. Shared Memory Segments are attached by means of the `shmat` system call and, by default, fall between the text segment and the private data segment. This means that the problem of private data segment expansion (by using the `sbrk` or `brk` system call) being restricted by an attached SMS does not exist on RISC-based systems. Programs should let the system default the attach address, whenever possible. For more details, see `shmop`(2) in the *ULTRIX Reference Pages*.

  Attach points for Shared Memory Segments (SMS) in the virtual address space of a process must be aligned on 4-Mbyte boundaries. Shared Memory Segments are attached by means of the `shmat` system call. Processes that permit the system to default the attach points will find that they are properly aligned. Processes that explicitly attach to a given address will find that the attach will succeed only if the given address is 4-Mbyte aligned or if the SHM_RND flag is set. Note that the latter case will cause the address to round down to a 4-Mbyte boundary. This restriction is imposed by hardware constraints. Programs should let the system default the attach address whenever possible. For more details, see `shmop`(2) in the *ULTRIX Reference Pages*.

- Floating

  There is no floating or double in the kernel. The FPU is assigned to a process, and the kernel manipulates scalars only. The `fixpoint.h` header file contains macros to convert an integer to its floating point format. User programs can use this.

- Float format

  VAX processors typically use D-float floating-point format. RISC processors use IEEE floating-point format (close to G-float). Thus, on RISC-based systems, there is greater range of floating numbers but less precision (fewer decimal places).

  Programs that need the extra precision of D-float and programs that need to be cognizant of the low-level format of floating point numbers will have problems. This will be true in RISC-based systems FORTRAN or any language that does floating-point work.

  In addition, there is no equivalent to D-float or H-float.

  Programs that incorrectly treat a float as a double, or vice versa, sometimes work on a VAX system in spite of the error. On a RISC system, this programming error causes incorrect results.

- Page size

  The page size on a RISC-based system is 4 Kbytes (4*1024) in contrast to 1 Kbyte on VAX-based system. Programs should use the `getpagesize` system call or include `vmmac.h` and use the macros for page size manipulations. For further information, see `getpagesize`(2) in the *ULTRIX Reference Pages*.

- `prof`

  The RISC version of `prof` is functionally different from the VAX version. For further information, see `prof`(1) in the *ULTRIX Reference Pages*.

- `ranlib`

  The `ranlib` command organizes archives of object files to allow faster linking. While this command still exists for RISC-based systems, it is a shell script to pass a flag to the `ar` librarian, which then performs the same function.

- `lint`

  Differences in the messages printed and the conditions checked exist between the RISC and VAX versions of `lint`. To build `lint` libraries, use one of the following:

  ```
  % lint –C libname myprog.c (VAX-based systems)
  % lint –c myprog.c (RISC-based systems)
  ```

  Note that the command line on RISC-based systems uses System V syntax.

- `nlist`

  Because RISC-based systems use Common Object File Format (COFF) for object files, this structure is different from that used on VAX-based systems. Programs with hard-coded initializations that assume the VAX `nlist` structure will have to be changed.

- Common Object File Format (COFF)

  RISC-based systems use the Common Object File Format (COFF) in its object files and load modules. Therefore, the following utilities that make use of the object file format have been replaced by their new versions: `nm` (has more symbol types), `dbx`, `as`, `ld`, `ar`, `size`, and `strip`.

- `nroff`

  The `nroff` command with the `-h` (tab) option produces different output on RISC systems than on VAX systems.

  The version of `nroff` on RISC-based systems uses ASCII device driver tables in contrast to the version on VAX systems, which uses device driver tables in the format of binary VAX object files.

  Note that this applies to `nroff` only. It does not apply to `troff`, as was previously documented. The `troff` command does not use device driver tables to format output.

- `a.out.h`

  The `a.out.h` header file does not include `exec.h` on RISC-based systems, as it does on VAX-based systems.

- `brk`

  On VAX-based systems, the `brk` program's virtual address space begins at zero. Text starts at zero and runs to &etext. Data then follows to &edata. The bss segment then follows to &end, and the rest is available for growth.

  On RISC-based systems, the virtual address space begins at 0x00400000. Text starts at 0x00400000 and runs to &etext. There is a gap to 0x10000000, where the data begins. Data runs to &edata, is followed by bss to &end, and the rest is available for growth.

  The major implication here is the interaction between `brk` and `getrlimit`. On VAX-based systems, doing a `getrlimit` for RLIMIT_DATA was a possible approximation for the maximum value that could be passed to `brk`. On RISC-based systems, the correct value is as follows:

  ```
  "the value returned by a getrlimit" +0x10000000;
  ```

  On VAX-based systems, the correct value is as follows:

  ```
  "the value returned by a getrlimit"+&etext.
  ```

  One solution is to use `sbrk`, not `brk`.

- `cc`

  The RISC C compiler is different from the VAX C compiler. The following differences should be noted:

  – This version of `cc` does not support the `const` keyword.

  – Pointers in RISC-based systems are unsigned; in VAX-based systems they are signed.

  – On RISC-based systems, you cannot dereference NULL pointers; includes arg to strlen.

  – RISC-based systems do not support `asm` in any form.

  – The RISC C compiler does not allow ''old-fashioned initialization.'' An example of this, which worked on VAX-based systems but gave a warning, and does not work on RISC-based systems is ''int i 0;''.

  – The `varargs` function is different on RISC-based systems. Any program that tries to walk the argument list by taking the address of an argument and incrementing it will not be successful, especially for double precision

arguments. Programs using the macros in `varargs.h` will work. Compiling with the `–varargs` option on RISC-based systems will attempt to detect nonportable code.

- The `setjmp/longjmp` buffer is larger on RISC-based systems. Programs with a hard-coded 10 word buffer will fail; programs that correctly include `<setjmp.h>` and declare a 'jmp_buf' will work correctly.
- The RISC C compiler has boundary alignment rules. User programs should only see this as a performance issue (the kernel does fix-ups). It is better, however, to align double-words, words, and half-words on natural boundaries. See `uac`(1) in the *ULTRIX Reference Pages*.
- Pointers cannot be used as the variable on `switch` statements on RISC-based systems.
- The RISC C compiler will not allow the same `.c` or `.o` file to be listed twice. It will generate doubly defined symbol errors; the VAX C compiler (`cc`) allowed this.
- On VAX-based systems, the `cc –L` option on the command line collectively affects `–l` options. On RISC-based systems, the `cc –L` option is seen strictly left to right. Therefore, if `–L` is supposed to affect `–l`, the `–L` must come first.
- On RISC-based systems, global symbols do not have an extra leading underscore. This mostly affects assembler programmers.
- The `–R` option (read-only text) is not supported.
- The `–Md` or `–Mg` options are not needed on RISC-based systems; the hardware has only one double precision format.
- RISC-based systems define a macro (for example, LANGUAGE_C) for the preprocessor that makes it possible to write multilingual include files.
- For `cpp` predefined symbols, `ultrix`, `unix`, and `bsd4_2` are defined on both RISC-based and VAX-based systems. On RISC-based systems, the equivalent predefined symbol of `vax` is `mips` and MIPSEL and `host_mips` are also defined.
- The following RISC C compiler options are not supported on VAX-based systems: `–P` (preprocess, produce .i); `–Wphase,opt`; RISC-based systems recognize the environment variables ROOTDIR and TMPDIR; `–cpp` and `–nocpp` (most useful for languages other than C); `–G` (relevant only to RISC architecture), `–j`, `–k`, and `–ko` (relevant only to RISC-based systems compiler design); `–std` (warn nonstd usage; `vcc` has `–V standard=portable`); `–volatile` and `–varargs` (modify compiler behavior in certain areas); `–V` (print versions); and `–EB` and `–EL`. For more information, see the *Guide to Languages and Programming*.

- Profiling on VAX-based systems has two levels that can be selected with the `–p` and `–pg` options. Profiling on RISC-based systems also has two levels that can be selected with the `–p` option or by running the post-processor `pixie` program. The RISC C compiler is not affected by either option; all work is done in the assembler or loader (or postprocessor).
- One level of optimization exists on VAX-based systems, which is off by default and enabled with the `–O` option. Five levels of optimization exist on RISC-

based systems. By default, the second level is used, which can be disabled with the –O0 option. The –O or –O2 options invoke something comparable to VAX-level optimization. There are rather more complex processes that can be used with –O3 and –O4 options. RISC-based systems also have the –Olimit option that allows optimization to be bypassed with overly complicated code sections.

- On both RISC-based and VAX-based systems, the –t and –B options specify passes and paths; however, the pass names for –t differ (there are more on RISC-based systems), and the semantics of –B belong to the –h option. The –B option is used to specify a command suffix instead. RISC-based systems also have the –H, –K, and –# options that are designed for compiler development work.
- Like optimization, RISC-based systems offer four levels for debugging information (controlled by the –g option). VAX-based systems have only two (on and off).
- If a global data item is used as if it were a code location (for example, if a data structure has the same name as a system call), an error message similar to the following will be printed at load time:

  ```
  /lib/libc.a(gethostent.o): jump relocation out-of-range, bad object
  file produced, can't jump from 0x4197a0 to 0x10008198 (stat)
  ```

  If this happens, you should change the name of the data structure (in this example, it was named `stat`).

D.2 Hints for Porting Software to RISC-Based Systems

The following are provided as helpful hints for porting software to a RISC-based system:

- NULL pointers

  On VAX-based systems, a NULL pointer can be dereferenced because page zero of user process space is mapped and valid. On RISC-based systems, however, you cannot dereference a NULL pointer without a segmentation violation. The pointer must be tested for non-NULL first.

- Alignment

  On VAX-based systems, short words (2 bytes) or long words (4 bytes) can be accessed on any byte boundary. On RISC-based systems, however, references must be ''naturally aligned.'' Short words (2 bytes) must be on an even byte boundary. Long words (4 bytes) must be accessed on a boundary evenly divisible by 4.

  Unaligned accesses in user programs cause a ''trap'' into the kernel (the system attempts to ''fixup'' the unaligned access). If the system is able to accomplish the fixup, a message is printed to the controlling tty (if there is one) stating at what pc the alignment error was encountered. If the system is not able to fixup the unaligned access, the process will be terminated with a SIGBUS (bus error) signal. Doing the fixups for the program does have a performance impact on the program. For further information, see `uac`(1) in the *ULTRIX Reference Pages*.

- Signed and unsigned pointers

  On VAX-based systems, if a pointer type function returns a –1 as an error status, the comparison "if (ptr < 0)" can be made, because pointers are signed values.

  On RISC-based systems, if a pointer type function returns a –1 as an error status, the comparison "if (ptr < 0)" will never be true, because pointers are unsigned values. (Thus, the compiler removes the code for the comparison.) The error returns are not caught. The comparison could be "if ((int)ptr < 0)" or "if (ptr == (char *) –1)".

- Variable number of arguments

  On RISC-based systems, a set of macros exists to declare formal parameters and access arguments in a list of a variable number of arguments. For further information, see the `/usr/include/varargs.h` or `/sys/h/varargs.h` files.

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-234-1998 using a 1200- or 2400-baud modem from anywhere in the USA, Canada, or Puerto Rico. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital Subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	—————	Local Digital subsidiary or approved distributor
Internal*	—————	SSB Order Processing - WMO/E15 *or* Software Supply Business Digital Equipment Corporation Westminster, Massachusetts 01473

* For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

ULTRIX/UWS
Release Notes
AA-ME85D-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

What would you like to see more/less of? _____

_____

_____

What do you like best about this manual? _____

_____

_____

What do you like least about this manual? _____

_____

_____

Please list errors you have found in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

What version of the software described by this manual are you using? _____

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Email _____ Phone _____

Do Not Tear – Fold Here and Tape

digital™

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST–CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZKO3–2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062–9987

Do Not Tear – Fold Here

Cut
Along
Dotted
Line

Reader's Comments

ULTRIX/UWS
Release Notes
AA-ME85D-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

What would you like to see more/less of? _____

_____

_____

What do you like best about this manual? _____

_____

_____

What do you like least about this manual? _____

_____

_____

Please list errors you have found in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

What version of the software described by this manual are you using? _____

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Email _____ Phone _____

Do Not Tear – Fold Here and Tape

digital™

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST–CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZKO3–2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062–9987

Do Not Tear – Fold Here

Cut
Along
Dotted
Line