ULTRIX-32
Guide to System
Configuration File Maintenance

Order No. AA-ME90A-TE

ULTRIX-32 Operating System, Version 3.0

Digital Equipment Corporation

This manual was written and produced by the ULTRIX Documentation Group in Nashua, New Hampshire.

Contents

About This Manual

Audience v

Organization v

Related Documents vi

Conventions vi

1 The System Configuration File

1.1 The System Configuration File 1-1

1.2 The Generic System Configuration File 1-1

1.3 The Generic System Configuration File Format 1-2

1.3.1 Global Definitions 1-2
1.3.2 System Image Definitions 1-8
1.3.3 Device Definitions 1-10

1.3.3.1 Adapter Specifications 1-10
1.3.3.2 Master Specifications 1-11
1.3.3.3 Controller Specifications 1-12
1.3.3.4 Device Specifications 1-14
1.3.3.5 Disk Specifications 1-15

1.3.4 Pseudodevice Definitions 1-16

1.4 Sample Generic Configuration File 1-18

1.5 System Configuration Files for Diskless Clients 1-24

1.5.1 Default Diskless Configuration File Naming Conventions 1-24
1.5.2 Diskless Default Configuration File Differences 1-25
1.5.3 Diskless Configuration File Use 1-25
1.5.4 Sample Default Diskless Configuration File 1-26

2 Building the Kernel

2.1 When To Build a New Kernel 2-1
2.2 Building a Kernel Automatically 2-2
2.2.1 Using the doconfig Program 2-2
2.2.2 Testing the New Kernel 2-6
2.3 Building a New Kernel Manually 2-6
2.3.1 Edit the Configuration File 2-7
2.3.2 Prepare the Directory for the Binary Files 2-7
2.3.3 Define the Code Dependencies 2-8
2.3.4 Compile and Load the Binary Files 2-9
2.3.5 Boot the New Kernel 2-10
2.4 Building the Kernel After a Capacity Upgrade Installation 2-11

A Device Mnemonics

Index

Examples

1-1: Sample Configuration File 1-18
1-2: Sample QEQDVAX.dlconf Configuration File 1-27
2-1: Sample doconfig Execution 2-4

Tables

A-1: Devices Supported by MAKEDEV A-2

About This Manual

This guide provides information on how to maintain the system configuration file and how to build a new kernel system image. This guide also explains how to build a new kernel automatically or manually.

Audience

The *ULTRIX–32 Guide to System Configuration File Maintenance* is written for the person responsible for managing and maintaining an ULTRIX system. It assumes that this individual is familiar with ULTRIX commands, the system configuration, the system's controller/drive unit number assignments and naming conventions, and an editor such as vi or ed. You do not need to be a programmer to use this guide.

Organization

This manual consists of two chapters, one appendix, and an index. The chapters are:

Chapter 1: The System Configuration File
Explains the format of the generic configuration file and provides a sample configuration file. This chapter also describes the default configuration files used in a diskless environment.

Chapter 2: Building the Kernel
Describes how to build a kernel either automatically or manually, and explains how to build a new kernel after a capacity upgrade installation.

Appendix A: Device Mnemonics
Lists the supported device mnemonics and explains how to obtain detailed reference page information on devices.

Related Documents

You should have the hardware documentation for your system and peripherals.

Conventions

The following conventions are used in this manual:

special	In text, each mention of a specific command, option, partition, pathname, directory, or file is presented in this type.
command(x)	In text, cross-references to the command documentation include the section number in the reference manual where the commands are documented. For example: See the cat(1) command. This indicates that you can find the material on the cat command in Section 1 of the reference pages.
literal	In syntax descriptions, this type indicates terms that are constant and must be typed just as they are presented.
italics	In syntax descriptions, this type indicates terms that are variable.
[ ]	In syntax descriptions, square brackets indicate terms that are optional.
. . .	In syntax descriptions, a horizontal ellipsis indicates that the preceding item can be repeated one or more times.
function	In function definitions, the function itself is shown in this type. The function arguments are shown in italics.
UPPERCASE	The ULTRIX system differentiates between lowercase and uppercase characters. Enter uppercase characters only where specifically indicated by an example or a syntax line.
example	In examples, computer output text is printed in this type.
example	In examples, user input is printed in this bold type.
%	This is the default user prompt in multiuser mode.
#	This is the default superuser prompt.
>>>	This is the console subsystem prompt.

. In examples, a vertical ellipsis indicates that not all of the
. lines of the example are shown.
.

<KEYNAME> In examples, a word or abbreviation in angle brackets indicates that you must press the named key on the terminal keyboard.

<CTRL/x> In examples, symbols like this indicate that you must hold down the CTRL key while you type the key that follows the slash. Use of this combination of keys may appear on your terminal screen as the letter preceded by the circumflex character. In some instances, it may not appear at all.

The System Configuration File 1

This chapter explains the contents and format of the generic configuration file. A sample generic configuration file is provided to illustrate how specific information defines the hardware, software, and system parameters. The chapter also contains a description of the default configuration files and their location in a diskless system environment.

1.1 The System Configuration File

The system configuration file describes how you want the configuration software to build the kernel. It identifies all of the device driver source code that needs to be compiled into the kernel, as well as a number of system parameters that affect how the kernel operates. The kernel is the system image that controls system scheduling, memory management, input and output services, device management, and organization of the file systems. Provided you have enough disk space, you can build more than one kernel.

Except for diskless systems, the system configuration file resides in /usr/sys/conf and has the same name as the system name (in uppercase letters) which was defined during the installation procedure. For example, if you named your system tucson during the installation procedure, then the system configuration file name will be /usr/sys/conf/TUCSON. The /usr/sys/conf directory also has a generic system configuration file that you can use to tailor other system configuration files to your hardware.

1.2 The Generic System Configuration File

This section describes the organization and entries of the GENERIC configuration file, /usr/sys/conf/GENERIC, which is a template that you can use to build other configuration files. In addition to the configuration file information contained in this chapter, the following information will help you tailor a configuration file to your system's hardware, options, and requirements:

- Section 4 of the ULTRIX Reference Pages contains definitions of configuration file entries and the syntax to define supported devices. Use the descriptions in Section 4 to determine the correct syntax for changes to the configuration file

- Appendix A provides information on the names of the device mnemonics supported by MAKEDEV

1.3 The Generic System Configuration File Format

All configuration files, including the generic configuration file, have four parts:

- Global definitions
- System image definitions
- Device definitions
- Pseudodevice definitions

Note

In some cases, the system parameters discussed in this section do not appear in the GENERIC configuration file. These parameters, as well as some of the arguments to the parameters, are described here because the parameters are used in some system configuration files.

1.3.1 Global Definitions

The global definitions parameters apply to all the kernels generated by the configuration file. Each global definition appears on a separate line in the configuration file. Each line represents a tunable system parameter and begins with one of these keywords:

```
machine
cpu
ident
timezone
maxusers
maxuprc
manuva
maxtsiz
physmem
dmmin
dmmax
```

```
smmin
smmax
smseg
smsmat
smbrk
processors
scs_sysid
options
```

The following paragraphs display the syntax and describe how and when to use each parameter:

machine *type*

This parameter defines the hardware; the argument *type* must be vax: For example:

```
machine vax
```

cpu "*type*"

This parameter defines the processor; the argument *type* must be enclosed in quotes. For example:

```
cpu "VAX780"
```

The GENERIC configuration file lists the cpu types by processor class. This is because in some cases, the processor names have been equivalenced in the configuration software. For instance, the "MVAX" entry applies to the MicroVAX II and VAXstation 2000 processors. The VAX3600 entry in the GENERIC configuration file applies to all of the MicroVAX 3000, VAX 3000, and VAXserver 3000 type processors. The VAX 8200 applies to the VAX 8200 processor.

- If you know your processor class, then you can use that as your configuration file entry.
- If you do not know your processor class, then you can use the exact processor name. For example:

```
VAX8800
VAX8820
VAX8700
VAX8600
VAX8550
VAX8530
VAX8500
VAX8350
```

```
VAX8300
VAX8200
VAX6210
VAX6220
VAX3600
VAX3500
VAX3400
VAX3300
VAX785
VAX780
VAX750
VAX420
MVAX
```

- You can specify more than one `cpu` *type* for a kernel that can be booted on multiple cpu's. However, a kernel for multiple processors means that during the configuration process, your system will build more capabilities than it needs. The result is that in most cases, your kernel will require more memory than a kernel for a single processor requires. It is also possible that under these conditions, your system will have to do more paging and swapping during daily operations, which will affect system performance.

ident *name*

This parameter defines the host machine for which you are creating the configuration file. The *name* argument is the system name which you specified during the installation procedure. Enter the name in upper cases letters.

```
ident TUCSON
```

This parameter ensures that all host-specific source code is compiled during the actual configuration process.

timezone *number* dst *x*

This parameter defines timezone information for your site. The installation procedure enters this value to your system configuration file according to information you supply during the installation or when you register a diskless client. The *number* argument identifies your time zone, measured by the number of hours west of Greenwich Mean Time: for example, Eastern Standard Time is five hours west of Greenwich Mean Time, and Pacific Standard Time is eight hours west. Negative numbers indicate hours east of GMT. The generic configuration file time zone entry is set to Eastern Daylight Savings Time (The entry is `timezone 5 dst`).

The argument dst indicates daylight savings time. During the installation procedure, you can include a number (x) to request a particular daylight savings time correction algorithm. The values are:

1 United States (the default value)

2 Australia

3 Western Europe

4 Central Europe

5 Eastern Europe

maxusers *number*

This parameter defines the maximum number of simultaneously active users allowed on your system. The *number* argument should be equal to or greater than the maximum number of users allowed by your license agreement.

The number in this field is used in the system algorithms to size a number of system data structures and to determine the amount of space allocated to system tables. One such table is the system process table, which is used to determine how many active processes can be running at one time.

The maxusers *number* also affects the number of mbuf pages that the kernel allocates based on increments of 32. For instance:

- If the maxusers number is less than 64, the kernel allocates 32 mbuf pages.
- If the maxusers number is between 64 and 95, the kernel allocates 64 mbuf pages.
- If the maxusers number is between 96 and 127, the kernel allocates 96 mbuf pages.
- If the maxusers number is between 128 and 159, the kernel allocates 128 mbuf pages.

maxuprc *number*

This parameter defines the the maximum number of processes one user can run simultaneously. The default maxuprc entry is 50.

maxuva *num*

This parameter defines the maximum aggregate number of user virtual address space in megabytes allowed by the system. The default value is 256 Mb.

maxsiz *num*

This parameter defines the largest text segment in megabytes allowed by the system. The default value is 12 Mb.

physmem *number*

This parameter defines an estimate of the amount of physical memory currently in the system in Mbytes. This *number* argument is not used to limit the amount of memory, rather, it is used by the system to size the system page table, so it should be greater than or equal to the amount of physical memory in the system.

dmmin *num1* and dmmax *num2*

The system satisfies requests for additional virtual memory using the values for dmmin and dmmax. A process is initially granted *num1* 512-byte blocks of virtual memory. The next time the process requests memory, the system grants twice as much (*num1* x 2). This allocation continues until the amount of memory granted is equal to *num2* blocks. After that, additional requests are satisfied with *num2* blocks of memory. The default value for *num1* is 32. The default value for *num2* is 1024. The *num1* and *num2* values should be a power of 2, and the *num2* value should be a multiple of *num1*. Otherwise, the system's behavior may be unpredictable.

The dmmin and dmmax parameters are used to size the maximum permitable process data segment. To get a maximum permitable process data segment of about 23 Mbytes, the dmmax value should be 1024. If you double the dmmax value to 2048, the process data segment size will be roughly 43 Mbytes. If you double 2048 to 4096, the process data segment size will be roughly 80 Mbytes. You should note that as the dmmax number increases, so does swap space fragmentation.

smmin *num*

This parameter defines the minimum number of 512 byte blocks of virtual memory at which a shared memory segment (SMS) may be sized. The default for smmin is 0 blocks. For more information see shmget(2) in the ULTRIX Reference Pages.

smmax *num*

This parameter defines the maximum number of 512 byte blocks of virtual memory at which a shared memory segment may be sized. The default for smmax is 256 blocks (128 Kbytes). For more information see shmget(2) in the ULTRIX Reference Pages.

smseg *num*

This parameter defines the maximum number of shared memory segments per process. The default value is 6. For more information see shmop(2) in the ULTRIX Reference Pages.

smbrk *num*

This parameter defines the default spacing between the end of a private data space of a process and the beginning of its shared data space in 512 byte blocks of virtual memory. This value is important because, once a process attaches shared memory, private data cannot grow past the beginning of shared data. The default for smbrk is 64 blocks (32 Kbytes). For more information on shared memory operations, see shmop(2) in the ULTRIX Reference Pages.

smsmat *num*

This parameter defines the highest attachable address in megabytes for shared memory segments. The default value is MAXDSIZE. For more information see shmop(2) in the ULTRIX Reference Pages.

processors *num*

This parameter defines the number of processors in the system.

scs_sysid *number*

This parameter identifies each host uniquely on the CI star cluster to the SCS subsystem. The *number* argument must be a unique identifier for each host. At installation, the system automatically generates this number, and puts it in the configuration file. If the system does not detect a CI at installation, it provides a default value of 1.

options *optionlist*

Although the options field allows optional code to be compiled into the system, you should leave the options as they appear in the generic configuration file. However, you can remove any of the options if they do not pertain to your site, or if your system is short on physical memory space. These are the possible values for *optionlist*:

EMULFLT	Emulates the floating point instruction set if it is not already present in the hardware.
INET	Provides internet communication protocols. The inet pseudodevice must also be set.
LAT	Allows you to access your machine from a local area terminal server on the Ethernet. The lta and lat pseudodevices must also be set.
DECNET	If the DECnet layered product is installed, this option must be set. The decnet pseudodevice must also be set.
QUOTA	Allows disk quotas to be set.

SYS_TRACE Enables the system call tracing capability. The sys_trace pseudodevice must also be set.

DLI Allows mop-mom activity to be invoked. The mop_mom command is usually included in the /etc/rc.local file as a background task to cause mop-mom to listen for down-line and up-line load requests over the network. The dli pseudodevice must also be set.

BSC Allows 2780/3780 emulation. The bsc pseudodevice must also be set.

RPC Remote Procedure Call facility

This option is necessary for RPC-based applications within the NFS file system. The rpc pseudodevice must also be set.

NFS Network File System

This option allows you to access the NFS protocol. It requires the RPC option. The nfs pseudodevice must also be set.

UFS ULTRIX File System

This option is the standard, local file system. If you do not use the NFS option, the UFS option must be set. If you do not specify this option, the system will be considered diskless. The ufs pseudodevice must also be set.

1.3.2 System Image Definitions

There is one system definition in the generic configuration file. However, you can change the definition or add more lines to the configuration file you are building to indicate that you want to generate more than one kernel. For each kernel you wish to generate, specify one line that begins with the keyword config. Each line can be used to define the root device, the swap area or areas, the dump area, and the argument processing area for system calls.

The general format of a line for any one kernel is:

```
config filename configuration-clauses
```

The *filename* argument is the name to be assigned to the file constituting the compiled kernel, or system image. The installation procedure assigns the name vmunix.

The *configuration-clauses* define the devices for the root file system, for the paging and swapping area, and for crash dumps. The *configuration-clauses* keywords are: root, swap, and dumps. The syntax and descriptions of these keywords are:

root [on] *device*

The installation procedure assigns partition a of the system disk to the root file system. You can change this assignment by editing the configuration file. For diskless clients, this entry is set to root on se0.

swap [on] *device* [and *device*] [size *x*] [boot]

The first *device* argument specifies the device and partition that you want the system to use for a paging and swapping area. The installation procedure assigns partition b of the system disk for the paging and swapping area. You can change this assignment by editing the configuration file.

The second *device* argument enables you to add another partition so that the kernel interleaves paging and swapping between the two partitions. To specify a second paging and swapping area, use the and clause with a device, a logical unit, and a partition name.

Use the size clause to specify a nonstandard partition size for one or more swap areas. The value of *x* represents the number of 512-byte sectors. A size larger than the associated disk partition is trimmed to the partition size. The default swap device is partition b of the device where the root is located.

If you specify swap on boot, the a partition of the booted device becomes the root, and swap space is assumed to be the b partition of the same device.

Example configuration file entries are:

```
swap on ra0b
swap on ra1h
swap on ra0b and ra1h
```

In the first example, the system swaps on partition b of the ra0 disk. In the second example, the system swaps on partition h of the ra1 disk. In the last example, the system swaps on partition b of the ra0 disk and partition h of the ra1 disk.

For diskless systems, if the swap file is remote, then you do not have to specify a swap device.

dumps [on] *device*

The *device* argument specifies the partition and the device where crash dumps are to be stored. The device that is specified must be on the same controller as the boot device. The default dump device

is the first swap device configured.

Usually, this entry is unnecessary in a diskless environment because the dms setup process specifies using the mop_mom command for dumping. However, customized diskless kernels can specify dumping to a disk. See mop_mom(8) for a description of this command. See the Guide to Diskless Management Services for more information on the creation of diskless environments.

1.3.3 Device Definitions

This section of the configuration file contains descriptions of each current or planned device on the system. You need to add definitions for devices that were not on the system at installation time. You may also want to delete device definitions for devices that have been removed from the hardware configuration.

Each line of this section of the file begins with one of these keywords:

adapter	Identifies a physical connection to a system bus such as VAXBI, MASSBUS, Q-bus, UNIBUS, MSI, IBUS, or CI.
master	A MASSBUS tape controller.
controller	Identifies either a physical or a logical connection with one or more slaves attached to it. Some examples are: uda, kdb, hsc, and uq.
device	An autonomous device which connects directly to a Q-bus, or to a UNIBUS, MASSBUS, IBUS, or VAXBI adapter (as opposed to a disk, for example, which connects through a disk controller).
disk	A disk drive connected to either a master or a controller.
tape	A tape drive connected to either a master or a controller.

The format of the information required for each of these types of devices varies, as described in the following sections:

1.3.3.1 Adapter Specifications – The adapters discussed in this section are the VAXBI, MASSBUS, UNIBUS, MSI, CI, IBUS, and Q-bus adapters. Each adapter is specified by its own format in the configuration file.

1. The format for VAXBI adapters is:

   adapter vaxbi*n* at nexus*?*

   The *n* is the unit number of the adapter. The question mark (*?*) allows the system to pick the appropriate NEXUS for you.

2. The format for MASSBUS adapters is:

   adapter mba*n* at nexus*?*

   The *n* is the unit number of the adapter. The question mark (*?*) allows the system to pick the appropriate NEXUS for you.

3. The format for IBUS adapters is:

   adapter ibus*n* at nexus*?*

4. The format for UNIBUS and Q-bus adapters is the same. Q-bus adapters are specific to MicroVAX– and VAXstation–type processors. The format is:

   adapter uba0 at nexus *?*

   The question mark (*?*) allows the system to pick the appropriate NEXUS for you.

5. The format for MSI adapters is:

   adapter msi0 at nexus*?*

   The question mark (*?*) allows the system to pick the appropriate NEXUS for you.

6. The formats for CI adapters are:

   adapter ci0 at nexus*?*

   adapter ci0 at vaxbi*?*

   The question mark (*?*) allows the system to pick the appropriate NEXUS or VAXBI for you.

1.3.3.2 Master Specifications – MASSBUS tape drives must be attached to a master. The format for specifying a master is:

master *dev*name at mba*m* drive*n*

dev The name of the tape device, such as ht0.

m The MASSBUS adapter number.

n The drive number.

For example:

```
master      ht0      at mba?      drive?
tape        tu0      at ht0       slave 0
tape        tu1      at ht0       slave 1
```

1.3.3.3 Controller Specifications – This section contains examples of the specifications for the various controllers. The controller examples are for MSCP, TMSCP and SCSI controllers. This section also defines the format for specifying other tape-to-disk interface controllers.

1. MSCP disk controllers

- For UNIBUS or Q-bus:

```
controller  uda0 at uba0
controller  uq0 at uda0 csr 0172150 vector uqintr
disk ra0 at uq0 drive 0
disk ra1 at uq0 drive 1
disk ra2 at uq0 drive 2
disk ra3 at uq0 drive 3
```

- For VAXBI:

```
controller kdb0 at vaxbi0 node?
controller uq0 at kdb0 vector uqintr
disk ra0 at uq0 drive 0
disk ra1 at uq0 drive 1
disk ra2 at uq0 drive 2
disk ra3 at uq0 drive 3
controller aio1 at vaxbi? node?
controller bvpssp0 at aio1 vector bvpsspintr
disk ra0 at bvpssp0 drive 0
```

- For VAX CI/HSC:

```
adapter ci0 at nexus?
adapter ci0 at vaxbi? node?
controller hsc0 at ci0 cinode0
disk ra0 at hsc0 drive0
```

- For MSI bus:

```
adapter msi0 at nexus?
controller dssc0 at msi0 msinode 0
disk ra0 at dssc0 drive 0
```

2. TMSCP tape controllers

- For UNIBUS or Q-bus:

```
controller klesiu0 at uba0
controller uq0 at klesiu0 csr 0174500 vector uqintr
tape tms0 at uq0 drive 0
```

- For VAXBI:

```
controller klesib0 at vaxbi0 node 0
controller uq0 at klesib0 vector uqintr
tape tms0 at uq0 drive 0
controller aie0 at vaxbi? node?
controller bvpssp0 at aie0 vector bvpsspintr
tape tms0 at bvpssp0 drive 0
```

- For MSI Bus:

```
adapter msi0 at nexus?
controller dssc0 at msi0 msinode0
tape tms0 at dssc0 drive 0
```

- For VAX CI/HSC:

```
adapter ci0 at nexus?
adapter ci0 at vaxbi? node?
controller hsc0 at ci0 cinode0
tape tms0 at hsc0 drive 0
```

3. SCSI controllers

- For disks:

```
adapter uba0 at nexus?
controller scsi0 at uba0 csr 0x200c0080 vector szintr
controller scsi1 at uba0 csr 0x200c0180 vector szintr
disk rz1 at scsi0 drive1
disk rz2 at scsi0 drive2
disk rz9 at scsi1 drive1
disk rz10 at scsi1 drive2
```

- For tapes:

```
adapter uba0 at nexus?
controller scsi0 at uba0 csr 0x200c0080 vector szintr
controller scsi1 at uba0 csr 0x200c0180 vector szintr
tape tz1 at scsi0 drive1
tape tz2 at scsi0 drive2
tape tz9 at scsi1 drive1
tape tz10 at scsi1 drive2
```

4. Other controllers

The format for controllers for the magnetic tape interface (ts) and the disk interface is:

```
controller dev at condev[ csr n ] vector vec
tape unit at  dev drive n
```

dev The device name and logical unit number of the controller.

condev The name and logical unit number of the device to which the controller is connected.

n For the controller, *n* represents the octal address of the control status register for the device. Note that the address needed here is a 16-bit address. This entry is not needed for the VAXBI. For the tape, *n* represents the logical name of the tape unit.

unit The unit number of the tape drive.

vec The address of any interrupt vector for the controller.

This example shows a sample entry for a TU80 or TSV05 (for MicroVAX) magnetic tape interface:

```
controller zs0 at uba0 csr 0172520 vector tsintr
tape       ts0 at zs0 drive 0
```

1.3.3.4 Device Specifications – The format for the hardware classified as devices is:

```
device dev  condev [csr n] [ flags f ]      vector  v1 ...
```

Tab characters are used to indicate continuation lines, if needed. The arguments are:

dev The device name and logical unit number of the device.

condev The name and logical unit number of the adapter or controller to which the device is connected.

n The octal address of the control status register for the device. The csr *n* option is not needed for VAXBI devices. A number used to convey information about the device to the device driver. The only flags for DIGITAL-supported devices are for line printers and communications multiplexers.

f The default page width for all DIGITAL line printers is 132 columns. To change the page width, use flags *f*, where *f* is a decimal number giving the desired width in columns. For example, to change to 80 columns, enter flags 80.

The DH, DZ, DMB, DHU, DMF, and DMZ communications multiplexers accept a hexadecimal flag value to specify any lines that should be treated as hardwired with carrier always present. The DHV-11, DZQ, and DZV serve the same function as the Q-bus. The format of the hexadecimal number is 0x*nn*, where *nn* is a hexadecimal number consisting of digits ranging from 0-9, a-f.

Because bits are numbered from right to left, setting bit 0 of the flag indicates that tty00 is hardwired, setting bit 1 of the flag indicates that tty01 is hardwired, and so forth. This example shows that tty02 is hardwired with carrier always present: flags 0x04

vl... The names of interrupt vector routines for the device driver.

The following example shows a sample device specification for the DEUNA 10-Mbyte Ethernet interface:

```
device de0 at uba0 csr 0174510 vector deintr
```

The following example shows a sample device specification for a DZ-11 communications multiplexer:

```
device dz0 at uba0 csr 0160100 flags 0xff vector dzrint dzxint
```

The following example shows a sample device specification for a DMB32 communications controller device:

```
device dmb0 at vaxbi2 node3 flags 0x00ff vector dmbsint dmbaint dmblint
```

1.3.3.5 Disk Specifications – The format for specifying disks is:

disk *dev* at *condev* drive *n*

dev The device name and logical unit number of the disk.

condev The name and logical unit number of the adapter or controller to which the disk is connected.

n The physical unit number of the disk. If your disk is on a MASSBUS device, you can specify a question mark (?) for *n*. A question mark (?) allows the system to assign the physical number to the disk for you.

Here is an example of a device specification for MSCP disks:

```
disk ra0 at uq0 drive 0
```

The MAKEDEV program allows you to make up to 32 RA units. You can have physical drive numbers (*n*) from 0 through 251, and logical drive numbers (*dev*) from 0 through 31. Number the drives consecutively. The physical drive number should correspond with its assigned logical drive number whenever possible, as shown in the preceding example. Therefore, the physical drive numbers from 32 through 251 are rarely used. Refer to MAKEDEV(8) for more information.

1.3.4 Pseudodevice Definitions

A pseudodevice is an operating system component for which there is no associated hardware such as a pseudoterminal or one of the various supported protocols. Pseudodevice definitions are needed in the config file so that the operating system will recognize these components.

Each pseudodevice definition line in the config file defines a driver for a particular pseudodevice. Each pseudodevice definition line begins with the keyword pseudodevice, followed by the pseudodevice name. The format is:

pseudo-device *name* [*max n*]

The *name* is the name of the pseudodevice. Configuration files can have the following pseudodevice names:

pty	For pseudoterminal support (default = 16, specify *max n* for more than 16).
inet	For DARPA internet protocols.
loop	For network loopback interface.
ether	For 10-Mbyte Ethernets.
lat	For local area terminal (LAT) protocols.
lta	For pseudoterminal driver (default = 16, specify *max n* for more than 16).
decnet	For support of DECNET, and is only required when the DECNET layered product is installed.
sys_trace	For support of the system call trace capability.
dli	For DLI support of mop_mom activity.
bsc	For support of 2780/3780 emulation. To work, the dpv0 or dup0 devices must be defined in the configuration file as shown in the example in Section 1.2.
rpc	For Remote Procedure Call facility.
nfs	For Network File System (NFS) protocol support.

ufs For local ULTRIX file system use.

scsnet For Systems Communications Services (SCS) network interface driver. See scs(4) in the ULTRIX Reference Pages for more information.

The optional *max n* argument lets you assign more pseudoterminals. By default, pty and lta are set to 16. If you need more pseudoterminals you must specify a *max n* value. For example:

```
pseudo-device pty 32
pseudo-device lta 32
```

1.4 Sample Generic Configuration File

Example 1–1 illustrates a typical generic configuration file. Be aware that the generic configuration file supplied with your system may be different from the one shown here.

Example 1-1: Sample Configuration File

```
#
# @(#)GENERIC4.1.1.18 (ULTRIX) 9/14/88
# GENERIC VAX
#
# Global Definitions
#
machine         vax
cpu       "VAX8800"
cpu       "VAX8600"
cpu       "VAX8200"
cpu       "VAX6210"
cpu       "VAX3600"
cpu       "VAX785"
cpu       "VAX780"
cpu       "VAX750"
cpu   "VAX420"
cpu       "MVAX"
ident           GENERIC
timezone        5 dst
maxusers        2
maxuprc         10
physmem         6
processors      1
scs_sysid       32
options         QUOTA
options         UFS
options         INET
options         EMULFLT
#
#
# System Image Definitions
#
#
config          vmunix          swap on boot
```

(continued on next page)

```
# Adapter Specifications
#
#
adapter       xmi0 at nexus?
adapter       vaxbi0 at nexus?
adapter       vaxbi1 at nexus?
adapter       vaxbi2 at nexus?
adapter       vaxbi3 at nexus?
adapter       vaxbi4 at nexus?
adapter       vaxbi5 at nexus?
adapter       vaxbi11 at nexus?
adapter       vaxbi12 at nexus?
adapter       vaxbi13 at nexus?
adapter       vaxbi14 at nexus?
adapter       mba0 at nexus?
adapter       mba0 at nexus?
adapter       mba1 at nexus?
adapter       mba0 at nexus?
adapter       mba1 at nexus?
adapter       mba2 at nexus?
adapter       mba3 at nexus?
adapter       uba0 at nexus?
adapter       uba1 at nexus?
adapter       uba2 at nexus?
adapter       uba3 at nexus?
adapter       uba4 at nexus?
adapter       uba5 at nexus?
adapter       uba6 at nexus?
adapter       ibus0 at nexus?
adapter       ibus1 at nexus?
adapter       ibus2 at nexus?
adapter       ibus3 at nexus?
adapter       ibus4 at nexus?
adapter       ibus5 at nexus?
adapter       ibus6 at nexus?
adapter       ibus7 at nexus?
adapter       msi0 at nexus?
adapter       ci0 at nexus?
adapter       ci0 at vaxbi? node?
#
#
# Controller Specifications
#
#
controller    hsc0 at ci0 cinode 0
controller    hsc1 at ci0 cinode 1
controller    hsc2 at ci0 cinode 2
controller    hsc3 at ci0 cinode 3
controller    hsc4 at ci0 cinode 4
controller    hsc5 at ci0 cinode 5
```

(continued on next page)

```
controller    hsc6 at ci0 cinode 6
controller    hsc7 at ci0 cinode 7
controller    dssc0 at msi0 msinode 0
controller    dssc0 at msi0 msinode 0
controller    dssc1 at msi0 msinode 1
controller    dssc2 at msi0 msinode 2
controller    dssc3 at msi0 msinode 3
controller    dssc4 at msi0 msinode 4
controller    dssc5 at msi0 msinode 5
controller    dssc6 at msi0 msinode 6
controller    dssc7 at msi0 msinode 7
controller    aio0 at vaxbi? node?
controller    aio1 at vaxbi? node?
controller    aie0 at vaxbi? node?
controller    aie1 at vaxbi? node?
controller    aie2 at vaxbi? node?
controller    aie3 at vaxbi? node?
controller    kdb0 at vaxbi? node?
controller    kdb1 at vaxbi? node?
controller    kdb2 at vaxbi? node?
controller    kdb3 at vaxbi? node?
controller    kdb4 at vaxbi? node?
controller    kdb5 at vaxbi? node?
controller    kdb6 at vaxbi? node?
controller    kdb7 at vaxbi? node?
controller    kdb8 at vaxbi? node?
controller    kdb9 at vaxbi? node?
controller    kdb10 at vaxbi? node?
controller    kdb11 at vaxbi? node?
controller    klesib0 at vaxbi? node?
controller    klesib1 at vaxbi? node?
controller    klesib2 at vaxbi? node?
controller    uda0 at uba?
controller    uda1 at uba?
controller    uda2 at uba?
controller    uda3 at uba?
controller    klesiu0 at uba?
controller    klesiu1 at uba?
controller    klesiu2 at uba?
controller    klesiu3 at uba?
controller    uq0 at uda0 csr 0172150 vector uqintr
controller    uq1 at uda1 csr 0172150 vector uqintr
controller    uq2 at uda2 csr 0172150 vector uqintr
controller    uq3 at uda3 csr 0172150 vector uqintr
controller    uq4 at kdb0 vector uqintr
controller    uq5 at kdb1 vector uqintr
controller    uq6 at kdb2 vector uqintr
controller    uq7 at kdb3 vector uqintr
```

(continued on next page)

```
controller   bvpssp0 at aio0 vector bvpsspintr
controller   bvpssp1 at aio1 vector bvpsspintr
controller   bvpssp2 at aie0 vector bvpsspintr
controller   uq0 at uda0 csr 0172150 vector uqintr
controller   uq1 at uda1 csr 0172150 vector uqintr
controller   uq2 at uda2 csr 0172150 vector uqintr
controller   uq3 at uda3 csr 0172150 vector uqintr
controller   uq16 at klesiu0  csr 0174500 vector uqintr
controller   uq17 at klesiu1  csr 0174500 vector uqintr
controller   uq18 at klesiu2  csr 0174500 vector uqintr
controller   uq19 at klesiu3  csr 0174500 vector uqintr
controller   uq20 at klesib0  vector uqintr
controller   uq21 at klesib1  vector uqintr
controller   scsi0 at uba0 csr 0x200c0080 vector szintr
controller   scsi1 at uba0 csr 0x200c0180 vector szintr
#
#
# Disk Specifications
#
#
disk     hp0  at mba? drive 0
disk     hp1  at mba? drive 1
disk     hp2  at mba? drive 2
disk     hp3  at mba? drive 3
disk     hp4  at mba? drive 4
disk     hp5  at mba? drive 5
disk     hp6  at mba? drive 6
disk     hp7  at mba? drive 7
disk     ra0  at uq0 drive 0
disk     ra1  at uq0 drive 1
disk     ra2  at uq0 drive 2
disk     ra3  at uq0 drive 3
disk     ra4  at uq0 drive 4
disk     ra5  at uq0 drive 5
disk     ra6  at uq0 drive 6
disk     rz0  at scsi0 drive0
disk     rz1  at scsi0 drive1
disk     rz2  at scsi0 drive2
disk     rz3  at scsi0 drive3
disk     rz4  at scsi0 drive4
disk     rz5  at scsi0 drive5
disk     rz6  at scsi0 drive6
disk     rz7  at scsi0 drive7
disk     rz8  at scsi1 drive0
disk     rz9  at scsi1 drive1
disk     rz10 at scsi1 drive2
disk     rz11 at scsi1 drive3
```

(continued on next page)

```
# Tape Specifications
#
#
tape      st0  at stc0 drive0
tape      ts0  at zs0 drive0
master    ht0  at mba? drive?
tape      tu0  at ht0 slave 0
tape      tu1  at ht0 slave 1
tape      tu2  at ht0 slave 2
tape      tu3  at ht0 slave 3
master    mt0  at mba? drive ?
tape      mu0  at mt0 slave 0
tape      mu1  at mt0 slave 1
tape      mu2  at mt0 slave 2
tape      mu3  at mt0 slave 3
tape      tms1 at mscp drive 1
tape      tms2 at mscp drive 2
tape      tms3 at mscp drive 3
tape      tms4 at mscp drive 4
tape      tms5 at mscp drive 5
tape      tms6 at mscp drive 6
tape      tms7 at mscp drive 7
tape      tz0  at scsi0 drive 0
tape      tz1  at scsi0 drive 1
tape      tz2  at scsi0 drive 2
tape      tz3  at scsi0 drive 3
tape      tz4  at scsi0 drive 4
tape      tz5  at scsi0 drive 5
tape      tz6  at scsi0 drive 6
tape      tz7  at scsi0 drive 7
tape      tz8  at scsi1 drive 8
tape      tz9  at scsi1 drive 9
tape      tz10 at scsi1 drive 10
tape      tz11 at scsi1 drive 11
tape      tz12 at scsi1 drive 12
tape      tz13 at scsi1 drive 13
tape      tz14 at scsi1 drive 14
tape      tz15 at scsi1 drive 15
#
#
# Workstation Specifications
#
#
device     qv0  at uba0 csr 0177200 flags 0x0f vector qvkint qvvint
device     qd0  at uba0 csr 0177400 flags 0x0f vector qddint qdaint qdiint
device     qd1  at uba0 csr 0177402 flags 0x0f vector qddint qdaint qdiint
device     sm0  at uba0 csr 0x200f0000 flags 0x0f vector smvint
device     sg0  at uba0 csr 0x3c000000 flags 0x0f vector sgaint sgfint
device     lx0  at vaxbi? node? vector lxbvpint
```

(continued on next page)

```
# Network Specifications
#
#
device        bvpni0   at aie0       vector bvpniintr
device        bvpni1   at aie1       vector bvpniintr
device        bvpni2   at aie2       vector bvpniintr
device        bvpni3   at aie3       vector bvpniintr
device        de0  at uba?       csr 0174510 vector deintr
device        de1  at uba?       csr 0174510 vector deintr
device        qe0  at uba0       csr 0174440 vector qeintr
device        se0  at uba0       csr 0x200e0000 vector seintr
device        ln0  at ibus0 vector lnintr
#
#
# Terminal and Printer Specifications
#
#
device     ss0  at uba? csr 0x200a0000 flags 0x0f vector ssrint ssxint
device     sh0  at uba0 csr 0x38000000 flags 0xff vector shrint shxint
device     lp0  at uba? csr 0177514 vector lpintr
device     dmb0 at vaxbi? node? flags 0xff vector dmbsint dmbaint dmblint
device     dmb1 at vaxbi? node? flags 0xff vector dmbsint dmbaint dmblint
device     dmb2 at vaxbi? node? flags 0xff vector dmbsint dmbaint dmblint
device     dmb3 at vaxbi? node? flags 0xff vector dmbsint dmbaint dmblint
device     dmb4 at vaxbi? node? flags 0xff vector dmbsint dmbaint dmblint
device     dmb5 at vaxbi? node? flags 0xff vector dmbsint dmbaint dmblint
device     dmb6 at vaxbi? node? flags 0xff vector dmbsint dmbaint dmblint
device     dmb7 at vaxbi? node? flags 0xff vector dmbsint dmbaint dmblint
device     dmb8 at vaxbi? node? flags 0xff vector dmbsint dmbaint dmblint
device     dmb9 at vaxbi? node? flags 0xff vector dmbsint dmbaint dmblint
#
#
# Pseudodevice Specifications
#
#
pseudo-device      ufs
pseudo-device      pty
pseudo-device      loop
pseudo-device      inet
pseudo-device      ether
pseudo-device      scsnet
```

1.5 System Configuration Files for Diskless Clients

This section describes the default system configuration files that the Diskless Management Services (dms) utility uses to establish diskless clients on a server. The section:

- Identifies the diskless configuration file naming conventions
- Explains some of the differences between the diskless configuration files and the configuration files on systems that have disks
- Describes how dms utility uses the diskless configuration files to configure diskless clients
- Provides a sample of a diskless configuration file

1.5.1 Default Diskless Configuration File Naming Conventions

The default diskless configuration files reside in /usr/var/diskless/defs, or in /var/diskless/defs if you did an advanced installation. In either case, the names of these files all end with .dlconf. The upper case letters that precede .dlconf identify the processor type. These upper case letters use the following conventions:

- The first two letters identify the Ethernet communications device. These letters agree with the corresponding communications device mnemonic, for example QE or SE.
- The second two letters identify the graphics device. These letters agree with the corresponding graphics device mnemonic, for example, QD, QV, SM, or SG. One exception is that non-graphic devices have the letters TE to denote a non-graphic terminal. Additionally, the second two letters can be followed by 2 to denote a dual-headed processor, for example QD2.
- The remaining three letters identify the architecture type, for example VAX.

An example of a diskless default configuration file name is QEQDVAX.dlconf.

The available default system configuration files and their corresponding systems are:

- QEQDVAX.dlconf – A VAXstation II/GPX that uses a VCB02 video subsystem (QDSS monochrome or color), or a VAXstation 3000 processor that uses a VCB02 video subsystem (QDSS color)
- QEQVVAX.dlconf – A VAXstation II processor that uses a VCB01 video subsystem (QVSS monochrome)
- QEQD2VAX.dlconf – A VAXstation II/GPX, dual-headed processor that has two VCB02 video subsystems (QDSS monochrome or color)

- QETEVAX.dlconf – A MicroVAX II, non-graphic device or a MicroVAX 3000 processor
- SESGVAX.dlconf – A VAXstation 2000 processor that uses a color monitor
- SESMVAX.dlconf – A VAXstation 2000 processor that uses a monochome monitor
- SETEVAX.dlconf – A MicroVAX 2000 non-graphic device
- LNTEVAX.dlconf – A VAX 3400 processor
- PVSGVAX.dlconf – A VAXstation 3100 processor that uses a color monitor
- PVSMVAX.dlconf – A VAXstation 3100 processor that uses a monochrome monitor

1.5.2 Diskless Default Configuration File Differences

This section lists some of the differences between the entries in these configuration files and the entries in configuration files for systems that have disks. These differences include the following:

- The timezone entry is not included in the diskless configuration file. The dms utility puts the server's timezone information into the client's configuration file automatically.
- The root entry in the diskless configuration file is always an Ethernet device such as qe0 or se0 depending on the processor type. This is because the client's root is on the server's system and can only be accessed over the network.
- By default, none of the diskless configuration files specify the xos pseudodevice. When installed into the diskless environment, the worksystem software puts this entry into the client's configuration file automatically.
- Two of the default configuration files (QEQDVAX.dlconf and QEQTEVAX.dlconf) specify more than one cpu. This is so the kernel can be booted on more than one processor type.

1.5.3 Diskless Configuration File Use

The dms utility uses the doconfig command to configure a diskless client on a server system. The form of the doconfig command that dms invokes is:

```
/etc/doconfig -c $i -p $INSDIR
```

The $i variable is the default configuration file name. The $INSDIR variable defines the full pathname of a diskless default root area.

Upon completion of a typical dms client installation, the diskless client will have access to a kernel that exists in one of two places depending on whether an advanced installation was performed. If an advanced installation was performed, the client will have access to a kernel on the server's disk in:

/var/diskless/dlenv*x*/root*x*.vax/usr/sys/*config_file_name*.dlconf/vmunix

If an advanced installation was not performed, then the client will have access to a kernel on the server's disk in:

/usr/var/diskless/dlenv*x*/root*x*.vax/usr/sys/*config_file_name*.dlconf/vmunix

In this syntax, *x* is a number denoting a particular diskless environment where *config_file_name*.dlconf coincides with one of the default configuration files listed in Section 1.3.1. An example pathname for a client who has installed a VAXstation II/GPX using an advanced installation is:

```
/var/diskless/dlenv0/root0.vax/usr/sys/QEQDVAX.dlconf/vmunix
```

This pathname structure enables many clients to share the same kernel. However, in some dms client installations, you may choose to set the diskless client up with its own kernel in its own root directory. The process of creating diskless clients or changing the location of a client's kernel is described in the Guide to Diskless Management Services.

Note

Never make changes to the supplied default configuration files. If you want to build a customized diskless system kernel, copy the default file to a separate area and make the changes to the copied version.

It is important to maintain the same default file names because the dms utility only builds kernels based on the default files supplied with the system.

1.5.4 Sample Default Diskless Configuration File

Example 1–2 shows a sample QEQDVAX.dlconf default configuration file. All of the configuration files have the same format as this one and except for the specified devices, are almost identical.

Refer to Section 1.1 for a detailed description of each of the entries.

Example 1-2: Sample QEQDVAX.dlconf Configuration File

```
# @(#)QEQDVAX.dlconf      3.7
machine         vax
ident           "QEQDVAX"
cpu             "MVAX"
cpu             "VAX3600"
maxusers        32
processors      1
maxuprc         25
physmem         4
timezone


options         INET
options         NFS
options         RPC
options         LAT
options         EMULFLT


config  vmunix     root on qe0

adapter  uba0  at nexus ?
device   qe0   at uba0 csr 0174440              vector qeintr
device   qd0   at uba0 csr 0177400 flags 0x0f vector qddint qdaint qdiint


pseudo-device pty
pseudo-device loop
pseudo-device ether
pseudo-device inet
pseudo-device nfs
pseudo-device rpc
pseudo-device lat
pseudo-device lta
```

Building the Kernel 2

This chapter describes how to build a kernel. There are three procedures from which to choose:

1. You can build a new kernel automatically using the doconfig command. Section 2.1 describes this procedure.

2. You can build the kernel manually following the steps listed in Section 2.2. If you opt to build the kernel manually, make sure you understand the contents and format of the configuration file. Chapter 1 describes this file.

3. You can build a kernel when you are performing a capacity upgrade installation. Section 2.3 describes this procedure.

Choose the procedure that best complements your experience and the needs of your particular installation.

2.1 When To Build a New Kernel

You need to build a new kernel after any of the following events:

- If you add a new device and its driver to your configuration. When you add a new device and device driver, you need to rebuild the kernel to include the specifications in the configuration file.
- If you delete a device and its driver from your configuration. When you delete a device and device driver from your configuration and edit the configuration file to include only the actual hardware and software at your installation, you need to rebuild the kernel to match this configuration.
- If you tune the operating system. When you alter the default configuration or change the original disk setup, you need to rebuild the kernel. For example, if you create swap areas on two disk drives, thereby modifying the original single swap area on disk, you need to rebuild the kernel.
- If you upgrade your system. For example, if you increase the log-in capacity on your system, you need to rebuild the kernel.

- If you add layered products. For example, if you add the DECnet facility, or any layered product to your configuration, you need to rebuild the kernel.

2.2 Building a Kernel Automatically

The ULTRIX software provides the /etc/doconfig program with which you build your kernel automatically. The program prompts you for information about your system configuration, generates the necessary files and directories, then automatically builds the new kernel. The following section describes this procedure.

Note

Be aware that the command line entry for the /etc/doconfig program differs for diskless systems. Refer to Chapter 1 and to the Guide to Diskless Management Services for information about the diskless client kernel.

2.2.1 Using the doconfig Program

When updating an existing configuration file or creating a new one with /etc/doconfig, the system must be operating the generic kernel, vmunix.

To use the /etc/doconfig program, follow these steps:

1. Log in as superuser (root). You must be superuser to execute the doconfig command.

2. Shut the system down to single-user mode by typing:

```
# shutdown +5  "Building a new kernel"
```

   Before building the kernel, you must be in single-user mode because when doconfig completes, it rearranges the previously-defined symbols.

3. Save the running vmunix as vmunix.old.

4. Move /genvmunix to /vmunix.

5. Reboot the system to single user mode.

6. Check file systems.

7. Mount the /usr file system.

8. Run the doconfig program by typing:

```
# /etc/doconfig
```

When the program finishes, it prints a message showing the path and location of the new vmunix.

9. Move /vmunix to /genvmunix.

10. Copy the new vmunix to /vmunix. (Make certain that you use the pathname for vmunix that the doconfig program printed when it finished executing.)

11. Reboot the system.

Refer to doconfig(8) in the ULTRIX Reference Pages for details on the command and its options.

Example 2–1 depicts a sample execution of the doconfig program. It demonstrates how doconfig works on a VAXstation II/GPX, dual-display system (qd0, qd1 devices).

- Entries in square brackets [] are the default values. When you run doconfig, press the RETURN key to select the default value. The example shows the default entries typed in for presentation purposes only.
- After you enter the system name and the date and time information, the doconfig program builds a configuration file. When doconfig completes the configuration file build process, it loads vmunix, rearranges the symbol table, and makes the special files for the system based on the configuration.

Example 2-1: Sample doconfig Execution

```
# /etc/doconfig

Type the name of your system using alphanumeric characters.
The first character must be a letter.  For example, tinker.

Type your system name:tinker

You typed tinker as the name of your system.
Is this correct? Type y or n [y]: y

*** SPECIFY THE DATE AND TIME ***

Enter the current date and time in this format:
yymmddhhmm. Use two digits for year (yy),
month (mm), day (dd), hour (hh), and minute (mm).
You type the time in 24-hour format. For example,
for 11:30 p.m. on May 14, 1987, the response
would be:

    8705142330

Type the date and time [no default]: 8705142330

*** SPECIFY THE TIME ZONE INFORMATION ***

Enter the time zone for your area, using the options
listed in this table:

          Time Zone          Options
        ------------------------------
           Eastern             e
           Central             c
           Mountain            m
           Pacific             p
           Greenwich           g
        ------------------------------

You can also enter the number of hours (-12 to 12) in time
west of Greenwich.

Type the time zone [no default]: p

Does your area alternate between Daylight Savings
and Standard time [yes] ?yes
```

(continued on next page)

```
Enter the geographic area for Daylight Savings Time,
using the options listed in this table:
       Geographic Area    Options
     ---------------------------------
            USA                 u
            Australia           a
            Eastern Europe      e
            Central Europe      c
            Western Europe      w
     ---------------------------------

Type the geographic area [u]: u

Tue May 10 12:29:00 EDT 1988

*** System Configuration Procedure ***


Configuration file complete.

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

*** PERFORMING SYSTEM CONFIGURATION ***
    working ..... Tue May 10 12:29:00 EDT 1988
    working ..... Tue May 10 12:31:01 EDT 1988
    working ..... Tue May 10 12:33:02 EDT 1988
    working ..... Tue May 10 12:35:03 EDT 1988
    working ..... Tue May 10 12:37:04 EDT 1988
    working ..... Tue May 10 12:39:05 EDT 1988
    working ..... Tue May 10 12:41:06 EDT 1988
    working ..... Tue May 10 12:43:07 EDT 1988
    working ..... Tue May 10 12:45:09 EDT 1988

*** DEVICE SPECIAL FILE CREATION ***
    working ..... Tue May  3 12:05:59 EDT 1988
    working ..... Tue May  3 12:08:00 EDT 1988
```

2.2.2 Testing the New Kernel

Upon completion of the automatic configuration process, you can test the new kernel that you have built by performing the following steps:

1. Save your original kernel:

```
#mv /vmunix /vmunix.old
```

2. Put the newly-created kernel in the root directory. For instance, to put the kernel created in example 2–1 into the root directory, you would type:

```
#mv /sys/TINKER/vmunix /vmunix
```

3. Reboot the system:

```
# /etc/reboot
```

   If you have any problems booting the new kernel, you can reboot the system using the original kernel that you copied to /vmunix.old. Refer to the Guide to System Shutdown and Startup for booting information.

2.3 Building a New Kernel Manually

You can build a new kernel manually in either single-user or multiuser mode. However, it is recommended that you build it in single-user mode so that users will not affect the process of rebuilding the kernel. You can shut down the system to single-user mode with the following command:

```
# shutdown +5 "Building a new kernel"
```

To build a new kernel manually in either single-user or multiuser mode, you must perform the following steps:

1. Edit the configuration file
2. Prepare the directory for the binary files
3. Define code dependencies
4. Compile and load the binary files
5. Boot the new kernel

Each of these steps is described in the following sections. You must follow these steps consecutively.

2.3.1 Edit the Configuration File

The configuration file resides in the directory /usr/sys/conf and has the same name as your system, but in uppercase letters. For example, if your system is named myvax, your configuration file is named /usr/sys/conf/MYVAX.

The configuration file is the file you copy and edit when you build a new kernel. This file includes definitions for all supported devices. The supported devices are listed in Appendix A.

Follow these steps to copy and then to edit the configuration file.

1. Log in to the system as superuser (root).
2. Change your working directory to /usr/sys/conf by typing:

```
# cd /usr/sys/conf
```

3. Make a backup copy of the original configuration file. To do this, copy the original configuration file to another file in the same directory. The name of the working configuration file should be the same as the original configuration file, with NEW. as a prefix. For example, if your configuration file is MYVAX, type:

```
# cp MYVAX NEW.MYVAX
```

4. Change the file access permissions (mode) of the working configuration file to permit the owner (superuser) to write to it. For example, if your working configuration file is named NEW.MYVAX, type:

```
# chmod +w NEW.MYVAX
```

5. Edit the working file. If your configuration file is named MYVAX, then NEW.MYVAX is the file you should edit. Add or delete the entries you want to change, using the format and rules described in Chapter 1:

```
# vi NEW.MYVAX
      .
      .
      .
```

2.3.2 Prepare the Directory for the Binary Files

The second step is to create a directory for building the binary files that are used to create the new kernel. Use the mkdir command to make an empty directory. Then run the config utility to place the necessary binary files in the directory. The config utility uses the name of the configuration file you edited in step 1.

To generate the new binary files:

1. Make sure your working directory is /usr/sys/conf. (You should be in this directory after editing the configuration file.)
2. Run the mkdir command to create a directory in the /usr/sys directory. The directory and configuration file names must be the same as your system name, but in uppercase letters. For example, if the name of your system is myvax, the command to create the appropriate directory is:

```
# mkdir ../NEW.MYVAX
```

3. Run the config utility with the name of the working configuration file you edited in Section 2.2.1. When the utility finishes, it displays a reminder message for you to do a make depend. This example shows the command for a system named myvax:

```
# config NEW.MYVAX

Don't forget to run "make depend"
```

2.3.3 Define the Code Dependencies

The third step is to define the code dependencies. The code dependencies determine which binary files are needed and how they are to be built, based on your kernel's configuration.

To define the code dependencies:

1. Change your working directory to the one you created in Section 2.2.2. For example, if your system is named myvax, type:

```
# cd /usr/sys/NEW.MYVAX
```

2. Execute the make command with the clean parameter. For example:

```
# make clean
```

   This command ensures that the /usr/sys/NEW.MYVAX directory contains only the required files for creating the kernel specified by the NEW.MYVAX configuration file.

3. Execute the make command with the depend parameter. For example:

```
# make depend
.
.
.
```

This command instructs make to build or rebuild the rules that it needs to recognize interdependencies in the system source code. Executing this command will ensure that any changes to the system source code will be recompiled the next time you run the make command. The make command modifies the makefile, appending the dependencies to the end of the file. After make successfully completes, it updates the makefile.

2.3.4 Compile and Load the Binary Files

The fourth step is to compile and load the new binary files using the makefile that you created when you defined the code dependencies (Section 2.2.3).

To compile and load the binary files:

1. Use the make command to produce a complete binary system image, the kernel. The kernel is stored in the current directory. The system responds by displaying a number of messages as it compiles and loads the binary files. When the make command completes, the system redisplays the system prompt. For example:

```
# make
      .
      .
      .
#
```

2. Save the original kernel in the root (/) directory in case your new kernel fails to work. If the new kernel fails, you can recover by booting from the original kernel. Boot instructions are in Section 2.2.5. Move the original kernel to another filename in the root directory. For example:

```
# mv /vmunix  /vmunix.old
```

3. The output of the make command is a kernel named vmunix in the current directory. Move this file to the root directory and then change its mode. For example:

```
# mv vmunix  /vmunix
# chmod 755 /vmunix
```

The original /vmunix file is replaced by the new vmunix file and is ready to be booted. The original /vmunix resides in /vmunix.old because you copied it there in step 2.

2.3.5 Boot the New Kernel

If you are in single-user mode, use the reboot command to boot the new kernel, /vmunix. To boot the new kernel, type:

```
# /etc/reboot
```

If you are in multiuser mode, use the shutdown command with the appropriate options to boot the new kernel.

```
# /etc/shutdown -r +5 "Rebooting new kernel"
```

In this example, the processor halts and then automatically reboots using the default boot device. The system boots the /vmunix image.

If the new kernel fails to boot or displays errors, you can recover by booting the original kernel (/vmunix.old) and running that kernel until you determine the cause of the problem.

If the new kernel runs but displays errors:

1. Shut the system down:

   ```
   # /etc/shutdown -h now
   ```

2. After the system is halted, boot the system using the conversational mode, as described in the Guide to System Shutdown and Startup. When the boot prompt appears, boot the old kernel using the name of the original kernel that you saved in Section 2.2.4.

   The following example shows how to reboot the old kernel in conversational mode on a VAX-11/780:

   ```
   >>> b ask

   Enter image name:vmunix.old
   ```

   In this example, the system boots the default system disk in conversational mode using the original system image which was renamed to vmunix.old. See the Guide to System Shutdown and Startup for processor-specific booting information and for information on booting in conversational mode.

2.4 Building the Kernel After a Capacity Upgrade Installation

After you have completed an ULTRIX operating system capacity upgrade installation, you need to build a new kernel. You should use the doconfig command to build the kernel.

The doconfig command asks you questions about your system, such as what time zone you are in, if you have daylight savings time, and so forth. It also shows you possible responses.

When doconfig asks if you want to edit the configuration file, type yes. The doconfig command then asks for the name of the editor you want to use. Once you are editing the configuration file, change the maxusers number to the new number of authorized users provided in your upgrade installation kit. For example, if your system currently has a maximum of 32 users, and you have an upgrade installation kit for 64 users, substitute the number 64 for 32. In this case, the new entry would read:

```
maxusers 64
```

Exit the editor and continue answering the doconfig utility prompts. Most of your answers will be no, unless you are adding or deleting devices.

After the doconfig utility completes, you can test the new kernel that you have built by performing the following steps:

1. Save your original kernel:

   ```
   #mv /vmunix /vmunix.old
   ```

2. Put the newly-created kernel in the root directory. For example, to put the kernel created in Example 2–1, you would type:

   ```
   #mv /sys/TINKER/vmunix /vmunix
   ```

3. Reboot the system:

   ```
   # /etc/reboot
   ```

   If you have any problems booting the new kernel, you can reboot the system using the original kernel that you copied to /vmunix.old. Refer to the Guide to System Shutdown and Startup for booting information. If you have any problems booting the newly-created kernel, you can boot the old kernel /vmunix.old and then try the capacity upgrade installation again.

   Also, if you have difficulties using the doconfig command, you can build the kernel manually using the steps described in this chapter.

Device Mnemonics A

This appendix identifies and defines the mnemonics that are used to attach any hardware or software device to your system. The mnemonics are used by the /dev/MAKEDEV shell script to create the character or block special files that represent each of the devices. The mnemonics also appear in the system configuration file as described in the Guide to System Configuration File Maintenance.

Table A–1 lists the mnemonics in seven categories: generic, consoles, disks, tapes, terminals, modems, and printers. The generic category lists the mnemonics of a general nature and includes memory, null, trace, and tty devices. The consoles category lists the system console devices that the ULTRIX operating system uses. The disks, tapes, terminals, modems, and printers categories identify the appropriate mnemonics for those devices.

The description heading in Table A–1 identifies the corresponding device name. It does not define the mnemonic's use. For detailed information on the use of each mnemonic in relation to both the MAKEDEV script and the system configuration file, refer to the reference pages in Section 4 of the ULTRIX Reference Pages. If on-line reference pages are available, you can also use the man command. For instance, if you enter at the system prompt:

```
# man ra
```

the system displays the reference page for the Mass Storage Control Protocol (MSCP) disk controller driver. Where appropriate, the SYNTAX section of the reference page defines the device's syntax as it appears, or should appear, in the config file. Refer to /dev/MAKEDEV for additional software device mnemonics that MAKEDEV uses. Refer to MAKEDEV(8) in the ULTRIX Reference Pages for a description of the MAKEDEV utility.

You should note that Table A–1 uses the convention of an asterisk (*) beside a mnemonic and a question mark (?) beside a device name to mean a variable number. The range of the variable number is dependent on the particular device.

Table A-1: Devices Supported by MAKEDEV

Category	Mnemonic	Description
Generic	boot*	Boot and std devices by cpu number; e.g., boot750
	mvax*	All MicroVAX setups; e.g., mvax2000
	vaxstation*	A VAXstation 2000 setup; e.g., vaxstation2000
	std	Standard devices below with all console subsystems:
	drum	Kernel drum device
	errlog	Error log device
	kUmem	Kernel Unibus/Q-bus virtual memory
	kmem	Virtual main memory
	mem	Physical memory
	null	A null device
	trace	A trace device
	tty	A tty device
	local	Customer specific devices
Consoles	console	System console interface
	crl	Console RL02 disk interface for VAX 86?0
	cs*	Console RX50 floppy interface for VAX 8??0
	ctu*	Console TU58 cassette interface for VAX 11/750
	cty*	Console extra serial line units for VAX 8??0
	cfl	Console RX01 floppy interface for 11/78?
	ttycp	Console line used as auxiliary terminal port
Disks	hp*	MASSBUS disk interface for RM?? drives
	ra*	UNIBUS/Q-bus/BI/HSC MSCP disk controller interface
	ese*	UNIBUS/Q-bus/BI/HSC MSCP electronic ESE20 disk
	rb*	UNIBUS IDC RL02 disk controller interface for RB?? drives
	rd*	VAXstation 2000 and MicroVAX 2000 RD type drives
	rz	SCSI disks (RZ22/RZ23/RZ55/RRD40)
	rk*	UNIBUS RK?? disk controller interface
	rl*	UNIBUS/Q-bus RL?? disk controller interface
	rx*	VAXstation 2000 and MicroVAX 2000 RX type drives
Tapes	mu*	TU78 MASSBUS magtape interface
	tms*	UNIBUS/Q-bus/BI/HSC TMSCP tape controller interface
	rv*	UNIBUS/Q-bus/BI/HSC TMSCP optical disk
	ts*	UNIBUS/Q-bus TS11/TS05/TU80 magtape interface
	tu*	TE16/TU45/TU77 MASSBUS magtape interface
	st*	VAXstation 2000 and MicroVAX 2000 TZK50 cartridge tape

Category	Mnemonic	Description
	tz*	SCSI tapes (TZ30/TZK50)
Terminals	cxa*	Q-bus cxa16
	cxb*	Q-bus cxb16
	cxy*	Q-bus cxt08
	dfa*	Q-bus DFA01 comm multiplexer
	dhq*	Q-bus DHQ11 comm multiplexer
	dhu*	UNIBUS DHU11 comm multiplexer
	dhv*	Q-bus DHV11 comm multiplexer
	dmb*	BI DMB32 comm multiplexer including dmbsp serial printer/plotter
	dhb*	BI DHB32 comm multiplexer
	dmf*	UNIBUS DMF32 comm multiplexer including dmfsp serial printer/plotter
	dmz*	UNIBUS DMZ32 comm multiplexer
	dz	UNIBUS DZ11 and DZ32 comm multiplexer
	sh*	MicroVAX 2000, 8 serial line expansion option
	ss*	VAXstation 2000 and MicroVAX 2000 basic 4 serial line unit
	dzq*	Q-bus DZQ11 comm multiplexer
	dzv*	Q-bus DZV11 comm multiplexer
	lta*	Sets of 16 network local area terminals (LAT)
	pty*	Sets of 16 network pseudoterminals
	qd*	Q-bus VCB02 (QDSS) graphics controller/console
	qv*	Q-bus VCB01 (QVSS) graphics controller/console
	sm*	VAXstation 2000 monochrome bitmap graphics/console
	sg*	VAXstation 2000 color bitmap graphics console
Modems	dfa*	DFA01 integral modem communications device.
Printers	dmbsp*	BI DMB32 serial printer/plotter
	dmfsp*	UNIBUS DMF32 serial printer/plotter
	lp*	UNIBUS LP11 parallel line printer
	lpv*	Q-bus LP11 parallel line printer

Index

C

capacity upgrade installation
rebuilding the kernel after, 2-11
configuration file (diskless)
available default files, 1-24
default, 1-24 to 1-27
description, 1-24 to 1-27
differences, 1-25
naming conventions, 1-24
sample configuration file, 1-27e
use, 1-25
configuration file (Generic)
example, 1-23
configuration file (system)
adapter specifications, 1-10 to 1-11
allocating virtual memory, 1-6
defined
device definition syntax, 1-10 to 1-16
device specifications, 1-14
disk specification syntax, 1-15
editing, 2-7
estimating physical memory, 1-6
example, 1-18
format, 1-1 to 1-17
global definition syntax, 1-2 to 1-8
identifying number of processors, 1-7
master specifications, 1-11 to 1-12
pseudodevice definition syntax, 1-16 to 1-17
configuration file (system) (cont.)
sizing process data segment, 1-6
specifying controllers, 1-12 to 1-14
specifying CPU, 1-3
specifying machine, 1-3
specifying maximum processes, 1-5
specifying maximum users, 1-5
specifying optional code, 1-7
specifying system name, 1-4
specifying time zone, 1-4
system image definition syntax, 1-8 to 1-10

D

daylight saving time
specifying, 1-4
doconfig command
building kernel automatically, 2-2

K

kernel
building, 2-1 to 2-11
building automatically, 2-2 to 2-6
building manually, 2-6 to 2-10
execution, 2-4e to 2-6e
when to rebuild, 2-1

M

MASSBUS adapter
configuration file format, 1-10
MASSBUS controller
specifying, 1-11
MSCP disk controller
specifying, 1-12e, 1-12e

P

printer
specifying line width, 1-14

Q

Q-bus adapter
configuration file format, 1-11

S

SCSI controller
specifying, 1-13e

T

time zone
specifying, 1-4
TMSCP tape controller
specifying, 1-13e

U

UNIBUS adapter
configuration file format, 1-11

V

VAX BI adapter
configuration file format, 1-10

HOW TO ORDER ADDITIONAL DOCUMENTATION

DIRECT TELEPHONE ORDERS

In Continental USA
and New Hampshire,
Alaska or Hawaii
call **800-DIGITAL**

In Canada
call **800-267-6215**

DIRECT MAIL ORDERS (U.S. and Puerto Rico*)

DIGITAL EQUIPMENT CORPORATION
P.O. Box CS2008
Nashua, New Hampshire 03061

DIRECT MAIL ORDERS (Canada)

DIGITAL EQUIPMENT OF CANADA LTD.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

DIGITAL EQUIPMENT CORPORATION
PSG Business Manager
c/o Digital's local subsidiary
or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473

*Any prepaid order from Puerto Rico must be placed
with the Local Digital Subsidiary:
809–754–7575

Reader's Comments

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement. _____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Documentation Manager
ULTRIX Documentation Group
ZKO3-3/X18
Spit Brook Road
Nashua, N.H.
03063

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line

Reader's Comments

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement. _____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Documentation Manager
ULTRIX Documentation Group
ZKO3-3/X18
Spit Brook Road
Nashua, N.H.
03063

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line