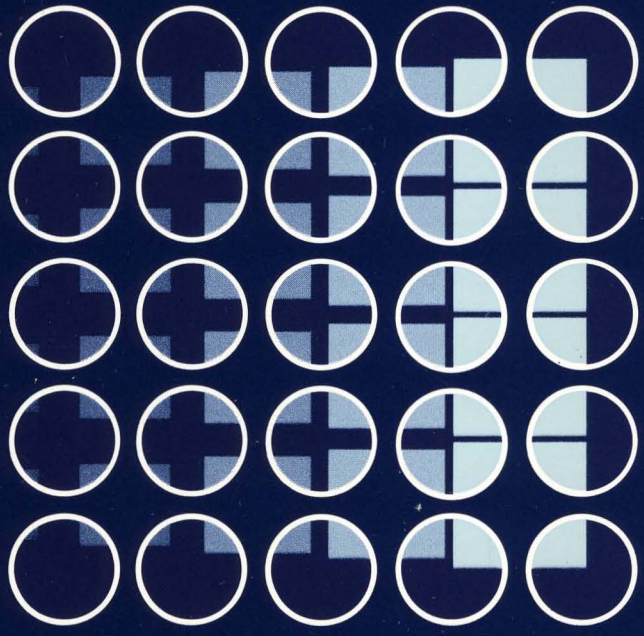
ULTRIX-32™
Programmer's Manual

ULTRIX-32™
Programmer's Manual
Sections 4 and 8

Order No. AA-BG56A-TE



digital software

ULTRIX-32™
Programmer's Manual
Sections 4 and 8

Order No. AA-BG56A-TE

---

digital equipment corporation, merrimack, new hampshire

First printing, May 1984

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

The postage-paid READER'S COMMENTS form on the last page of this document requests your critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	ULTRIX-32
DECUS	UNIBUS
MASSBUS	VAX
PDP	VMS
ULTRIX	VT
ULTRIX-11	digital™

UNIX is a trademark of AT&T Bell Laboratories.

Information herein is derived from copyrighted material as permitted under a license agreement with AT&T Bell Laboratories.

This software and documentation is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California. We acknowledge the Electrical Engineering and Computer Sciences Departments at the Berkeley Campus of the University of California for their role in its development.

This software and documentation is based in part on the Fourth Berkeley Software Distribution under license from The Regents of the University of California. We acknowledge the following individuals for their role in its development:

Eric Allman, Ken Arnold, Ozalp Babaoglu, Scott B. Baden, Jerry Berkman, John Breedlove, Earl T. Cohen, Robert P. Corbett, Mike Curry, Steve Feldman, Tom Ferrin, John Foderaro, Susan L. Graham, Charles Haley, Robert R. Henry, Andy Hertzfeld, Mark Horton, S.C. Johnson, William Joy, Howard Katseff, Peter Kessler, Jim Kleckner, J.E. Kulp, James Larus, Kevin Layer, Mike Lesk, Steve Levine, Jeff Levinsky, Louise Madrid, M. Kirk McKusick, Colin L. McMaster, Mikey Olson, Geoffrey Peck, Ed Pelegri-Llopart, Rob Pike, Dave Presotto, John F. Reiser, Asa Romberger, Bill Rowan, Jeff Schreibman, Eric P. Scott, Greg Shenaut, Eric Shienbrood, Kurt Shoens, Keith Sklower, Helge Skrivervik, Al Stanberger, Ken Thompson, Michael C. Toy, Richard Tuck, Bill Tuthill, Mike Urban, Edward Wang, David Wasley, Joseph Weizenbaum, Jon L. White, Glenn Wichman, Niklaus Wirth.

ULTRIX-32 Documentation Set

1. Organization

The ULTRIX-32 documentation set is organized into four separate binders. The documentation in each binder is so organized to better meet the needs of three separate audiences in the performance of their respective tasks. The ULTRIX-32 documentation set comprises:

Programmer's Manual Binder 1 – General Users

Section 1 – Commands
Documentation for all user-invoked programs (commands)

Section 6 – Games
Documentation for all user-invoked game programs

Programmer's Manual Binder 2 – Programmers

Section 2 – System Calls
Documentation for the system calls (entries into the kernel)

Section 3 – Subroutines
Documentation for the library subroutines

Section 5 – File Formats and Conventions
Documentation for the output and system file structures

Section 7 – Macro Packages and Language Conventions
Documentation for miscellaneous information

Programmer's Manual Binder 3A – System Managers

Installation Guide
Documentation for installing an ULTRIX-32 system

Building an ULTRIX-32 System with the Config Program
Documentation for configuring an executable kernel image

4.2BSD Line Printer Spooler
Documentation for installing the line printer spooling system

Sendmail Installation and Operations Guide
Documentation for installing and operating the sendmail system

UUCP Installation and Administration
: Documentation for installing and administering the uucp system

Programmer's Manual Binder 3B -- System Managers

Guidelines for System Management
: Documentation for maintaining an installed ULTRIX-32 system

Section 4 – Special Files
: Documentation for the special files (I/O devices and drivers)

Section 8 – Maintenance Commands
: Documentation for the system maintenance programs (commands)

2. Man(1) Format

Each numbered section in Binder 1, Binder 2, and Binder 3B contains an introduction and separate entries that correspond to those created by the man(1) command. Each entry, following the order in their respective binders, describes a user command or game; a system call, library subroutine, file structure, or macro package; and a special file or maintenance command.

Regardless of their respective binder, all entries have a single, consistent format. First, the header provides information that quickly identifies the entry: name and section number (enclosed in parentheses). For example, AARDVARK(6) identifies the aardvark(6) game (Binder 1). Then, the listed subsections provide specific information about the entry. Although each entry lists only those subsections that are applicable, seven main subsections may be used. Finally, the footer provides paging information: section and consecutive page number. For example, the footers for section 6 (Binder 1) are 6-1 through 6-35.

The seven main subsections are:

NAME
: This subsection lists the exact name and a short description of its function.

SYNTAX
: This subsection lists the complete syntax. Boldface indicates literals. A minus sign (-) indicates command options. Ellipses (...) indicate that the preceding argument may be repeated. Square brackets [ ] indicate optional arguments.

DESCRIPTION
: This subsection provides a detailed description of function and background.

FILES
: This subsection lists those related files that either are part of or are used during execution.

DIAGNOSTICS
This subsection lists those diagnostic messages that may be produced. Since most self-explanatory messages are not listed, this subsection is not comprehensive.

RESTRICTIONS
This subsection lists those restrictions that are known to apply.

SEE ALSO
This subsection lists the names of the related entries and other documentation.

3. Conventions

The following conventions apply specifically to these documents:

Installation Guide (Binder 3A)
Building an ULTRIX-32 System with the Config Program (Binder 3A)
UUCP Installation and Administration (Binder 3A)
Guidelines for System Management (Binder 3B)

bold	Literals are printed in **bold type**. Literals frequently indicate a specific command option and should be entered exactly as printed.
case	The ULTRIX-32 system differentiates between uppercase and lowercase. Therefore, enter uppercase only where specifically indicated by an example or the command syntax.
color	Examples are printed in color. Examples represent command sequences or information that the user enters from the terminal.
<CTRL/*X*>	Terminal control characters are represented by <CTRL/*X*>, where *X* is a single character. To generate a terminal control characters, hold down the CTRL key while entering the character.
<DELETE>	The DELETE key or ERASE character is represented by <DELETE>.
italics	Substitutable parameters are printed in *italics*.
<RETURN>	The RETURN key is printed as <RETURN>. To invoke a command, enter the command sequence and depress the RETURN key.

#	The superuser prompt (normally a #) is displayed at the console when the system is in single-user mode or at a terminal when the superuser is logged in.
>>>	The console subsystem prompt is represented by three right angle brackets, >>>. For further information about console commands, read the *VAX Hardware Manual.*

NAME

intro – introduction to special files and hardware support

DESCRIPTION

This section describes the special files, related driver functions, and networking support available in the system. In this part of the manual, the SYNTAX section of each configurable device gives a sample specification for use in constructing a system description for the *config*(8) program. The DIAGNOSTICS section lists messages which may appear on the console and in the system error log */usr/adm/messages* due to errors in device operation.

This section contains both devices which may be configured into the system, "4" entries, and network related information, "4N", "4P", and "4F" entries; The networking support is introduced in *intro*(4N).

VAX DEVICE SUPPORT

This section describes the hardware supported on the DEC VAX-11. Software support for these devices comes in two forms. A hardware device may be supported with a character or block *device driver*, or it may be used within the networking subsystem and have a *network interface driver*. Block and character devices are accessed through files in the file system of a special type; c.f. *mknod*(8). Network interfaces are indirectly accessed through the interprocess communication facilities provided by the system; see *socket*(2).

A hardware device is identified to the system at configuration time and the appropriate device or network interface driver is then compiled into the system. When the resultant system is booted, the autoconfiguration facilities in the system probe for the device on either the UNIBUS or MASSBUS and, if found, enable the software support for it. If a UNIBUS device does not respond at autoconfiguration time it is not accessible at any time afterwards. To enable a UNIBUS device which did not autoconfigure, the system will have to be rebooted. If a MASSBUS device comes "on-line" after the autoconfiguration sequence it will be dynamically autoconfigured into the running system.

The autoconfiguration system is described in *autoconf*(4). VAX specific device support is described in "4V" entries. A list of the supported devices is given below.

SEE ALSO

intro(4), intro(4N), autoconf(4), config(8)

LIST OF DEVICES

The devices listed below have driver supplied in this version of the system. Devices are indicated by their functional interface.

acc	ACC LH/DH IMP communications interface
ad	Data translation A/D interface
css	DEC IMP-11A communications interface
ct	C/A/T phototypesetter
de	DEC DEUNA communications controller
dh	DH-11 emulators, terminal multiplexor

dmc	DEC DMC-11/DMR-11 point-to-point communications device
dmf	DEC DMF-32 terminal multiplexor
dn	DEC DN-11 autodialer interface
dz	DZ-11 terminal multiplexor
ec	3Com 10Mb/s Ethernet controller
en	Xerox 3Mb/s Ethernet controller (obsolete)
kg	KL-11/DL-11W line clock
fl	VAX-11/780 console floppy interface
hk	RK6-11/RK06 and RK07 moving head disk
hp	MASSBUS disk interface (with RP06, RM03, RM05, etc.)
ht	TM03 MASSBUS tape drive interface (with TE-16, TU-45, TU-77)
hy	DR-11B or GI-13 interface to an NSC Hyperchannel
ik	Ikonas frame buffer graphics device interface
il	Interlan 10Mb/s Ethernet controller
lp	LP-11 parallel line printer interface
mt	TM78 MASSBUS tape drive interface
pcl	DEC PCL-11 communications interface
ps	Evans and Sutherland Picture System 2 graphics interface
rx	DEC RX02 floppy interface
tm	TM-11/TE-10 tape drive interface
ts	TS-11 tape drive interface
tu	VAX-11/730 TU58 console cassette interface
uda	DEC UDA-50 disk controller
un	DR-11W interface to Ungermann-Bass
up	Emulex SC-21V UNIBUS disk controller
ut	UNIBUS TU-45 tape drive interface
uu	TU58 dual cassette drive interface (DL11)
va	Benson-Varian printer/plotter interface
vp	Versatec printer/plotter interface
vv	Proteon proNET 10Mb/s ring network interface

STATUS

INTRO(4) currently is not supported by Digital Equipment Corporation.

NAME

networking – introduction to networking facilities

SYNTAX

#include <sys/socket.h>
#include <net/route.h>
#include <net/if.h>

DESCRIPTION

This section briefly describes the networking facilities available in the system. Documentation in this part of section 4 is broken up into three areas: *protocol-families*, *protocols*, and *network interfaces*. Entries describing a protocol-family are marked "4F", while entries describing protocol use are marked "4P". Hardware support for network interfaces are found among the standard "4" entries.

All network protocols are associated with a specific *protocol-family*. A protocol-family provides basic services to the protocol implementation to allow it to function within a specific network environment. These services may include packet fragmentation and reassembly, routing, addressing, and basic transport. A protocol-family may support multiple methods of addressing, though the current protocol implementations do not. A protocol-family is normally comprised of a number of protocols, one per *socket*(2) type. It is not required that a protocol-family support all socket types. A protocol-family may contain multiple protocols supporting the same socket abstraction.

A protocol supports one of the socket abstractions detailed in *socket*(2). A specific protocol may be accessed either by creating a socket of the appropriate type and protocol-family, or by requesting the protocol explicitly when creating a socket. Protocols normally accept only one type of address format, usually determined by the addressing structure inherent in the design of the protocol-family/network architecture. Certain semantics of the basic socket abstractions are protocol specific. All protocols are expected to support the basic model for their particular socket type, but may, in addition, provide non-standard facilities or extensions to a mechanism. For example, a protocol supporting the SOCK STREAM abstraction may allow more than one byte of out-of-band data to be transmitted per out-of-band message.

A network interface is similar to a device interface. Network interfaces comprise the lowest layer of the networking subsystem, interacting with the actual transport hardware. An interface may support one or more protocol families, and/or address formats. The SYNTAX section of each network interface entry gives a sample specification of the related drivers for use in providing a system description to the *config*(8) program. The DIAGNOSTICS section lists messages which may appear on the console and in the system error log */usr/adm/messages* due to errors in device operation.

PROTOCOLS

The system currently supports only the DARPA Internet protocols fully. Raw socket interfaces are provided to IP protocol layer of the DARPA Internet, to the IMP link layer (1822), and to Xerox PUP-1 layer operating on top of 3Mb/s Ethernet interfaces. Consult the appropriate manual pages in this section for more information regarding the support for

each protocol family.

ADDRESSING

Associated with each protocol family is an address format. The following address formats are used by the system:

```
#define   AF_UNIX          1     /* local to host (pipes, portals) */
#define   AF_INET          2     /* internetwork: UDP, TCP, etc. */
#define   AF_IMPLINK       3     /* arpanet imp addresses */
#define   AF_PUP           4     /* pup protocols: e.g. BSP */
```

ROUTING

The network facilities provided limited packet routing. A simple set of data structures comprise a "routing table" used in selecting the appropriate network interface when transmitting packets. This table contains a single entry for each route to a specific network or host. A user process, the routing daemon, maintains this data base with the aid of two socket specific *ioctl*(2) commands, SIOCADDRT and SIOCDELRT. The commands allow the addition and deletion of a single routing table entry, respectively. Routing table manipulations may only be carried out by super-user.

A routing table entry has the following form, as defined in *<net/route.h>*;

```
struct rtentry {
        u_long      rt_hash;
        struct      sockaddr rt_dst;
        struct      sockaddr rt_gateway;
        short       rt_flags;
        short       rt_refcnt;
        u_long      rt_use;
        struct      ifnet_*rt ifp;
};
```

with *rt_flags* defined from,

```
#define   RTF_UP           0x1   /* route usable */
#define   RTF_GATEWAY      0x2   /* destination is a gateway */
#define   RTF_HOST         0x4   /* host entry (net otherwise) */
```

Routing table entries come in three flavors: for a specific host, for all hosts on a specific network, for any destination not matched by entries of the first two types (a wildcard route). When the system is booted, each network interface autoconfigured installs a routing table entry when it wishes to have packets sent through it. Normally the interface specifies the route through it is a "direct" connection to the destination host or network. If the route is direct, the transport layer of a protocol family usually requests the packet be sent to the same host specified in the packet. Otherwise, the interface may be requested to address the packet to an entity different from the eventual recipient (i.e. the packet is forwarded).

Routing table entries installed by a user process may not specify the hash, reference count, use, or interface fields; these are filled in by the routing routines. If a route is in use when it is deleted (*rt refcnt* is non-zero), the resources associated with it will not be reclaimed until further references to it are released.

The routing code returns EEXIST if requested to duplicate an existing entry, ESRCH if requested to delete a non-existant entry, or ENOBUFS if insufficient resources were available to install a new route.

User processes read the routing tables through the */dev/kmem* device.

The *rt use* field contains the number of packets sent along the route. This value is used to select among multiple routes to the same destination. When multiple routes to the same destination exist, the least used route is selected.

A wildcard routing entry is specified with a zero destination address value. Wildcard routes are used only when the system fails to find a route to the destination host and network. The combination of wildcard routes and routing redirects can provide an economical mechanism for routing traffic.

INTERFACES

Each network interface in a system corresponds to a path through which messages may be sent and received. A network interface usually has a hardware device associated with it, though certain interfaces such as the loopback interface, *lo*(4), do not.

At boot time each interface which has underlying hardware support makes itself known to the system during the autoconfiguration process. Once the interface has acquired its address it is expected to install a routing table entry so that messages may be routed through it. Most interfaces require some part of their address specified with an SIOCSIFADDR ioctl before they will allow traffic to flow through them. On interfaces where the network-link layer address mapping is static, only the network number is taken from the ioctl; the remainder is found in a hardware specific manner. On interfaces which provide dynamic network-link layer address mapping facilities (e.g. 10Mb/s Ethernets), the entire address specified in the ioctl is used.

The following *ioctl* calls may be used to manipulate network interfaces. Unless specified otherwise, the request takes an *ifrequest* structure as its parameter. This structure has the form

```
struct    ifreq {
          char      ifr_name[16];              /* name of interface (e.g. "ec0") */
          union {
                    struct    sockaddr ifru_addr;
                    struct    sockaddr ifru_dstaddr;
                    short     ifru_flags;
          } ifr_ifru;
#define ifr_addr  ifr_ifru.ifru_addr  /* address */
#define ifr_dstaddr       ifr ifru.ifru_dstaddr       /* other end of p-to-p link */
#define ifr_flags  ifr_ifru.ifru_flags  /* flags */
```

```
};
```

SIOCSIFADDR
: Set interface address. Following the address assignment, the "initialization" routine for the interface is called.

SIOCGIFADDR
: Get interface address.

SIOCSIFDSTADDR
: Set point to point address for interface.

SIOCGIFDSTADDR
: Get point to point address for interface.

SIOCSIFFLAGS
: Set interface flags field. If the interface is marked down, any processes currently routing packets through the interface are notified.

SIOCGIFFLAGS
: Get interface flags.

SIOCGIFCONF
: Get interface configuration list. This request takes an *ifconf* structure (see below) as a value-result parameter. The *ifc_len* field should be initially set to the size of the buffer pointed to by *ifc_buf*. On return it will contain the length, in bytes, of the configuration list.

```
/*
 * Structure used in SIOCGIFCONF request.
 * Used to retrieve interface configuration
 * for machine (useful for programs which
 * must know all networks accessible).
 */
struct  ifconf {
        int     ifc_len;            /* size of associated buffer */
        union {
                caddr_t  ifcu_buf;
                struct   ifreq *ifcu_req;
        } ifc_ifcu;
#define ifc_buf  ifc_ifcu.ifcu_buf   /* buffer address */
#define ifc_req  ifc_ifcu.ifcu_req   /* array of structures returned */
};
```

SEE ALSO

socket(2), ioctl(2), intro(4), config(8), routed(8C)

STATUS

INTRO(4N) currently is not supported by Digital Equipment Corporation.

NAME

acc – ACC LH/DH IMP interface

SYNTAX

pseudo-device imp
device acc0 at uba0 csr 167600 vector accrint accxint

DESCRIPTION

The *acc* device provides a Local Host/Distant Host interface to an IMP. It is normally used when participating in the DARPA Internet. The controller itself is not accessible to users, but instead provides the hardware support to the IMP interface described in *imp*(4). When configuring, the *imp* pseudo-device must also be included.

DIAGNOSTICS

acc%d: not alive. The initialization routine was entered even though the device did not autoconfigure. This indicates a system problem.

acc%d: can't initialize. Insufficient UNIBUS resources existed to initialize the device. This is likely to occur when the device is run on a buffered data path on an 11/750 and other network interfaces are also configured to use buffered data paths, or when it is configured to use buffered data paths on an 11/730 (which has none).

acc%d: imp doesn't respond, icsr=%b. The driver attempted to initialize the device, but the IMP failed to respond after 500 tries. Check the cabling.

acc%d: stray xmit interrupt, csr=%b. An interrupt occurred when no output had previously been started.

acc%d: output error, ocsr=%b, icsr=%b. The device indicated a problem sending data on output.

acc%d: input error, csr=%b. The device indicated a problem receiving data on input.

acc%d: bad length=%d. An input operation resulted in a data transfer of less than 0 or more than 1008 bytes of data into memory (according to the word count register). This should never happen as the maximum size of a host-IMP message is 1008 bytes.

STATUS

ACC(4) currently is not supported by Digital Equipment Corporation.

NAME

ad – Data Translation A/D converter

SYNTAX

device ad0 at uba0 csr 0170400 vector adintr

DESCRIPTION

Ad provides the interface to the Data Translation A/D converter. This is **not** a real-time driver, but merely allows the user process to sample the board's channels one at a time. Each minor device selects a different A/D board.

The driver communicates to a user process by means of ioctls. The AD_CHAN ioctl selects which channel of the board to read. For example,

```
chan = 5; ioctl(fd, AD_CHAN, &chan);
```

selects channel 5. The AD_READ ioctl actually reads the data and returns it to the user process. An example is

```
ioctl(fd, AD_READ, &data);
```

FILES

/dev/ad

DIAGNOSTICS

None.

STATUS

AD(4) currently is not supported by Digital Equipment Corporation.

NAME

arp – Address Resolution Protocol

SYNTAX

pseudo-device ether

DESCRIPTION

ARP is a protocol used to dynamically map between DARPA Internet and 10Mb/s Ethernet addresses on a local area network. It is used by all the 10Mb/s Ethernet interface drivers and is not directly accessible to users.

ARP caches Internet-Ethernet address mappings. When an interface requests a mapping for an address not in the cache, ARP queues the message which requires the mapping and broadcasts a message on the associated network requesting the address mapping. If a response is provided, the new mapping is cached and any pending messages are transmitted. ARP itself is not Internet or Ethernet specific; this implementation, however, is. ARP will queue at most one packet while waiting for a mapping request to be responded to; only the most recently "transmitted" packet is kept.

ARP watches passively for hosts impersonating the local host (i.e. a host which responds to an ARP mapping request for the local host's address) and will, optionally, periodically probe a network looking for impostors.

DIAGNOSTICS

duplicate IP address!! sent from ethernet address: %x %x %x %x %x %x . ARP has discovered another host on the local network which responds to mapping requests for its own Internet address.

SEE ALSO

ec(4), il(4)

STATUS

ARP(4P) currently is not supported by Digital Equipment Corporation.

NAME

autoconf – diagnostics from the autoconfiguration code

DESCRIPTION

When UNIX bootstraps it probes the innards of the machine it is running on and locates controllers, drives, and other devices, printing out what it finds on the console. This procedure is driven by a system configuration table which is processed by *config*(8) and compiled into each kernel.

Devices in NEXUS slots are normally noted, thus memory controllers, UNIBUS and MASSBUS adaptors. Devices which are not supported which are found in NEXUS slots are noted also.

MASSBUS devices are located by a very deterministic procedure since MASSBUS space is completely probe-able. If devices exist which are not configured they will be silently ignored; if devices exist of unsupported type they will be noted.

UNIBUS devices are located by probing to see if their control-status registers respond. If not, they are silently ignored. If the control status register responds but the device cannot be made to interrupt, a diagnostic warning will be printed on the console and the device will not be available to the system.

A generic system may be built which picks its root device at boot time as the "best" available device (MASSBUS disks are better than SMD UNIBUS disks are better than RK07's; the device must be drive 0 to be considered.) If such a system is booted with the RB ASKNAME option of (see *reboot*(2)), then the name of the root device is read from the console terminal at boot time, and any available device may be used.

SEE ALSO

intro(4), config(8)

DIAGNOSTICS

cpu type %d not configured. You tried to boot UNIX on a cpu type which it doesn't (or at least this compiled version of UNIX doesn't) understand.

mba%d at tr%d. A MASSBUS adapter was found in tr%d (the NEXUS slot number). UNIX will call it mba%d.

%d mba's not configured. More MASSBUS adapters were found on the machine than were declared in the machine configuration; the excess MASSBUS adapters will not be accessible.

uba%d at tr%d. A UNIBUS adapter was found in tr%d (the NEXUS slot number). UNIX will call it uba%d.

dr32 unsupported (at tr %d). A DR32 interface was found in a NEXUS, for which UNIX does not have a driver.

ci unsupported (at tr %d). A CI interface was found in a NEXUS, for which UNIX does not have a driver.

mcr%d at tr%d. A memory controller was found in tr%d (the NEXUS slot number). UNIX will call it mcr%d.

5 mcr's unsupported. UNIX supports only 4 memory controllers per cpu.

mpm unsupported (at tr%d). Multi-port memory is unsupported in the sense that UNIX does not know how to poll it for ECC errors.

%s%d at mba%d drive %d. A tape formatter or a disk was found on the MASSBUS; for disks %s%d will look like "hp0", for tape formatters like "ht1". The drive number comes from the unit plug on the drive or in the TM formatter (**not** on the tape drive; see below).

%s%d at %s%d slave %d. (For MASSBUS devices). Which would look like "tu0 at ht0 slave 0", where **tu0** is the name for the tape device and **ht0** is the name for the formatter. A tape slave was found on the tape formatter at the indicated drive number (on the front of the tape drive). UNIX will call the device, e.g., **tu0.**

%s%d at uba%d csr %o vec %o ipl %x. The device %s%d, e.g. dz0 was found on uba%d at control-status register address %o and with device vector %o. The device interrupted at priority level %x.

%s%d at uba%d csr %o zero vector. The device did not present a valid interrupt vector, rather presented 0 (a passive release condition) to the adapter.

%s%d at uba%d csr %o didn't interrupt. The device did not interrupt, likely because it is broken, hung, or not the kind of device it is advertised to be.

%s%d at %s%d slave %d. (For UNIBUS devices). Which would look like "up0 at sc0 slave 0", where **up0** is the name of a disk drive and **sc0** is the name of the controller. Analogous to MASSBUS case.

STATUS

AUTOCONF(4) currently is not supported by Digital Equipment Corporation.

NAME

bk – line discipline for machine-machine communication (obsolete)

SYNTAX

pseudo-device bk

DESCRIPTION

This line discipline provides a replacement for the old and new tty drivers described in *tty*(4) when high speed output to and especially input from another machine is to be transmitted over a asynchronous communications line. The discipline was designed for use by the Berkeley network. It may be suitable for uploading of data from microprocessors into the system. If you are going to send data over asynchronous communications lines at high speed into the system, you must use this discipline, as the system otherwise may detect high input data rates on terminal lines and disables the lines; in any case the processing of such data when normal terminal mechanisms are involved saturates the system.

The line discipline is enabled by a sequence:

```
#include <sgtty.h>
int ldisc = NETLDISC, fildes; ...
ioctl(fildes, TIOCSETD, &ldisc);
```

A typical application program then reads a sequence of lines from the terminal port, checking header and sequencing information on each line and acknowledging receipt of each line to the sender, who then transmits another line of data. Typically several hundred bytes of data and a smaller amount of control information will be received on each handshake.

The old standard teletype discipline can be restored by doing:

```
ldisc = OTTYDISC;
ioctl(fildes, TIOCSETD, &ldisc);
```

While in networked mode, normal teletype output functions take place. Thus, if an 8 bit output data path is desired, it is necessary to prepare the output line by putting it into RAW mode using *ioctl*(2). This must be done **before** changing the discipline with TIOCSETD, as most *ioctl*(2) calls are disabled while in network line-discipline mode.

When in network mode, input processing is very limited to reduce overhead. Currently the input path is only 7 bits wide, with newline the only recognized character, terminating an input record. Each input record must be read and acknowledged before the next input is read as the system refuses to accept any new data when there is a record in the buffer. The buffer is limited in length, but the system guarantees to always be willing to accept input resulting in 512 data characters and then the terminating newline.

User level programs should provide sequencing and checksums on the information to guarantee accurate data transfer.

SEE ALSO

tty(4)

DIAGNOSTICS

None.

STATUS

BK(4) currently is not supported by Digital Equipment Corporation.

NAME

cons – VAX-11 console interface

DESCRIPTION

The console is available to the processor through the console registers. It acts like a normal terminal, except that when the local functions are not disabled, control-P puts the console in local console mode (where the prompt is ">>>"). The operation of the console in this mode varies slightly per-processor.

On an 11/780 you can return to the conversational mode using the command "set t p" (set terminal program) if the processor is still running or "continue" if it is halted. The latter command may be abbreviated "c". If you hit the break key on the console, then the console will go into ODT (console debugger mode). Hit a "P" (upper-case letter p) to get out of this mode.

On an 11/750 or an 11/730 the processor is halted whenever the console is not in conversational mode, and typing "C" returns to conversational mode. When in console mode on an 11/750 which has a remote diagnosis module, a ^D will put you in remote diagnosis mode, where the prompt will be "RDM>". The command "ret" will return from remote diagnosis mode to local console mode.

With the above proviso's the console works like any other UNIX terminal.

FILES

/dev/console

SEE ALSO

tty(4), reboot(8)
VAX Hardware Handbook

DIAGNOSTICS

None.

STATUS

CONS(4) is supported by Digital Equipment Corporation.

NAME

css – DEC IMP-11A LH/DH IMP interface

SYNTAX

pseudo-device imp
device css0 at uba0 csr 167600 flags 10 vector cssrint cssxint

DESCRIPTION

The *css* device provides a Local Host/Distant Host interface to an IMP. It is normally used when participating in the DARPA Internet. The controller itself is not accessible to users, but instead provides the hardware support to the IMP interface described in *imp*(4). When configuring, the *imp* pseudo-device is also included.

DIAGNOSTICS

css%d: not alive. The initialization routine was entered even though the device did not autoconfigure. This is indicates a system problem.

css%d: can't initialize. Insufficient UNIBUS resources existed to initialize the device. This is likely to occur when the device is run on a buffered data path on an 11/750 and other network interfaces are also configured to use buffered data paths, or when it is configured to use buffered data paths on an 11/730 (which has none).

css%d: imp doesn't respond, icsr=%b. The driver attempted to initialize the device, but the IMP failed to respond after 500 tries. Check the cabling.

css%d: stray output interrupt csr=%b. An interrupt occurred when no output had previously been started.

css%d: output error, ocsr=%b icsr=%b. The device indicated a problem sending data on output.

css%d: recv error, csr=%b. The device indicated a problem receiving data on input.

css%d: bad length=%d. An input operation resulted in a data transfer of less than 0 or more than 1008 bytes of data into memory (according to the word count register). This should never happen as the maximum size of a host-IMP message is 1008 bytes.

STATUS

CSS(4) currently is not supported by Digital Equipment Corporation.

NAME

ct – phototypesetter interface

SYNTAX

device ct0 at uba0 csr 0167760 vector ctintr

DESCRIPTION

This provides an interface to a Graphic Systems C/A/T phototypesetter. Bytes written on the file specify font, size, and other control information as well as the characters to be flashed. The coding is not described here.

Only one process may have this file open at a time. It is write-only.

FILES

/dev/cat

SEE ALSO

troff(1)
Phototypesetter interface specification

DIAGNOSTICS

None.

STATUS

CT(4) currently is not supported by Digital Equipment Corporation.

NAME

de – DEUNA 10 Mb/s Ethernet interface

SYNTAX

device de0 at uba0 csr 0174510 vector deintr

DESCRIPTION

The *de* interface provides access to a 10 Mb/s Ethernet network through the DEC DEUNA controller.

The host's Internet address is specified at boot time with an SIOCSIFADDR ioctl. The *de* interface employs the address resolution protocol described in *arp*(4P) to dynamically map between Internet and Ethernet addresses on the local network.

The interface normally tries to use a "trailer" encapsulation to minimize copying data on input and output. This may be disabled, on a per-interface basis, by setting the IFF_NOTRAILERS flag with an SIOCSIFFLAGS ioctl.

DIAGNOSTICS

There are many diagnostic error messages that can occur. These messages normally contain the relevant information that the deuna provides. If, for example the ethernet cable is shorted a message will be printed in which the registers will indicate a lost carrier.

SEE ALSO

intro(4N), inet(4F), arp(4P)

STATUS

DE(4) is supported by Digital Equipment Corporation.

NAME

dh – DH-11/DM-11 communications multiplexer

SYNTAX

device dh0 at uba0 csr 0160020 vector dhrint dhxint
device dm0 at uba0 csr 0170500 vector dmintr

DESCRIPTION

A dh-11 provides 16 communication lines; dm-11's may be optionally paired with dh-11's to provide modem control for the lines.

Each line attached to the DH-11 communications multiplexer behaves as described in *tty*(4). Input and output for each line may independently be set to run at any of 16 speeds; see *tty*(4) for the encoding.

Bit *i* of flags may be specified for a dh to say that a line is not properly connected, and that the line should be treated as hard-wired with carrier always present. Thus specifying "flags 0x0004" in the specification of dh0 would cause line ttyh2 to be treated in this way.

The dh driver normally uses input silos and polls for input at each clock tick (10 milliseconds) rather than taking an interrupt on each input character.

FILES

/dev/tty[hi][0-9a-f]
/dev/ttyd[0-9a-f]

SEE ALSO

tty(4)

DIAGNOSTICS

dh%d: NXM. No response from UNIBUS on a dma transfer within a timeout period. This is often followed by a UNIBUS adapter error. This occurs most frequently when the UNIBUS is heavily loaded and when devices which hog the bus (such as rk07's) are present. It is not serious.

dh%d: silo overflow. The character input silo overflowed before it could be serviced. This can happen if a hard error occurs when the CPU is running with elevated priority, as the system will then print a message on the console with interrupts disabled. If the Berknet is running on a *dh* line at high speed (e.g. 9600 baud), there is only 1/15th of a second of buffering capacity in the silo, and overrun is possible. This may cause a few input characters to be lost to users and a network packet is likely to be corrupted, but the network will recover. It is not serious.

STATUS

DH(4) currently is not supported by Digital Equipment Corporation.

NAME

dmc – DEC DMC-11/DMR-11 point-to-point communications device

SYNTAX

device dmc0 at uba0 csr 167600 vector dmcrint dmcxint

DESCRIPTION

The *dmc* interface provides access to a point-to-point communications device which runs at either 1 Mb/s or 56 Kb/s. DMC-11's communicate using the DEC DDCMP link layer protocol.

The *dmc* interface driver also supports a DEC DMR-11 providing point-to-point communication running at data rates from 2.4 Kb/s to 1 Mb/s. DMR-11's are a more recent design and thus are preferred over DMC-11's.

The host's address must be specified with an SIOCSIFADDR ioctl before the interface will transmit or recive any packets.

DIAGNOSTICS

dmc%d: bad control %o. A bad parameter was passed to the *dmcload* routine.

dmc%d: unknown address type %d. An input packet was received which contained a type of address unknown to the driver.

DMC FATAL ERROR 0%o.

DMC SOFT ERROR 0%o.

dmc%d: af%d not supported. The interface was handed a message which has addresses formatted in an unsuitable address family.

SEE ALSO

intro(4N), inet(4F)

STATUS

DMC(4) is supported by Digital Equipment Corporation.

NAME

dmf – DMF-32, terminal multiplexor

SYNTAX

device dmf0 at uba? csr 0170000
vector dmfsrint dmfsxint dmfdaint dmfdbint dmfrint dmfxint dmflint

DESCRIPTION

The *dmf* device provides 8 lines of asynchronous serial line support with full modem control on two lines only. (The DMF-32 provides other services, but these are not supported by the driver.)

Each line attached to a DMF-32 serial line port behaves as described in *tty*(4). Input and output for each line may independently be set to run at any of 16 speeds; see *tty*(4) for the encoding.

Bit *i* of flags may be specified for a *dmf* to to say that a line is not properly connected, and that the line should be treated as hard-wired with carrier always present. Thus specifying "flags 0x0004" in the specification of *dmf*0 would cause line ttyh2 to be treated in this way.

The *dmf* driver normally uses input silos and polls for input at each clock tick (10 milliseconds).

FILES

/dev/tty[hi][0-9a-f]
/dev/ttyd[0-9a-f]

SEE ALSO

tty(4)

DIAGNOSTICS

dmf%d: NXM line %d. No response from UNIBUS on a dma transfer within a timeout period. This is often followed by a UNIBUS adapter error. This occurs most frequently when the UNIBUS is heavily loaded and when devices which hog the bus (such as rk07's) are present. It is not serious.

dmf%d: silo overflow. The character input silo overflowed before it could be serviced. This can happen if a hard error occurs when the CPU is running with elevated priority, as the system will then print a message on the console with interrupts disabled. If the Berknet is running on a *dh* line at high speed (e.g. 9600 baud), there is only 1/15th of a second of buffering capacity in the silo, and overrun is possible. This may cause a few input characters to be lost to users and a network packet is likely to be corrupted, but the network will recover. It is not serious.

dmfsrint.
dmfsxint.
dmfdaint.
dmfdbint.
dmflint.
One of the unsupported parts of the dmf interrupted; something is amiss, check your

interrupt vectors for a conflict with another device.

STATUS

DMF(4) is supported by Digital Equipment Corporation.

NAME

dn – DN-11 autocall unit interface

SYNTAX

device dn0 at uba? csr 0160020 vector dnintr

DESCRIPTION

The *dn* device provides an interface through a DEC DN-11 (or equivalent such as the Able Quadracall) to an auto-call unit (ACU). To place an outgoing call one forks a sub-process which opens the appropriate call unit file, */dev/cua?* and writes the phone number on it. The parent process then opens the corresponding modem line */dev/cul?*. When the connection has been established, the open on the modem line, */dev/cul?* will return and the process will be connected. A timer is normally used to timeout the opening of the modem line.

The codes for the phone numbers are:

0-9	dial 0-9
*	dial * (':' is a synonym)
#	dial # (';' is a synonym)
–	delay 20 milliseconds
<	end-of-number ('e' is a synonym)
=	delay for a second dial tone ('w' is a synonym)
f	force a hangup of any existing connection

The entire telephone number must be presented in a single *write* system call.

By convention, even numbered call units are for 300 baud modem lines, while odd numbered units are for 1200 baud lines. For example, */dev/cua0* is associated with a 300 baud modem line, */dev/cul0*, while */dev/cua1* is associated with a 1200 baud modem line, */dev/cul1*. For devices such as the Quadracall which simulate multiple DN-11 units, the minor device indicates which outgoing modem to use.

FILES

/dev/cua?	call units
/dev/cul?	associated modem lines

SEE ALSO

tip(1C)

DIAGNOSTICS

Two error numbers are of interest at open time.

[EBUSY] The dialer is in use.

[ENXIO] The device doesn't exist, or there's no power to it.

STATUS

DN(4) currently is not supported by Digital Equipment Corporation.

NAME

drum – paging device

DESCRIPTION

This file refers to the paging device in use by the system. This may actually be a subdevice of one of the disk drivers, but in a system with paging interleaved across multiple disk drives it provides an indirect driver for the multiple drives.

FILES

/dev/drum

RESTRICTIONS

Reads from the drum are not allowed across the interleaving boundaries. Since these only occur every .5Mbytes or so, and since the system never allocates blocks across the boundary, this is usually not a problem.

STATUS

DRUM(4) currently is not supported by Digital Equipment Corporation.

NAME

dz – DZ-11 communications multiplexer

SYNTAX

device dz0 at uba0 csr 0160100 vector dzrint dzxint

DESCRIPTION

A dz-11 provides 8 communication lines with partial modem control, adequate for UNIX dialup use. Each line attached to the DZ-11 communications multiplexer behaves as described in *tty*(4) and may be set to run at any of 16 speeds; see *tty*(4) for the encoding.

Bit *i* of flags may be specified for a dz to say that a line is not properly connected, and that the line should be treated as hard-wired with carrier always present. Thus specifying "flags 0x04" in the specification of dz0 would cause line tty02 to be treated in this way.

The dz driver normally uses its input silos and polls for input at each clock tick (10 milliseconds) rather than taking an interrupt on each input character.

FILES

/dev/tty[0-9][0-9]
/dev/ttyd[0-9a-f] dialups

SEE ALSO

tty(4)

DIAGNOSTICS

dz%d: silo overflow. The 64 character input silo overflowed before it could be serviced. This can happen if a hard error occurs when the CPU is running with elevated priority, as the system will then print a message on the console with interrupts disabled. If the Berknet is running on a *dz* line at high speed (e.g. 9600 baud), there is only 1/15th of a second of buffering capacity in the silo, and overrun is possible. This may cause a few input characters to be lost to users and a network packet is likely to be corrupted, but the network will recover. It is not serious.

STATUS

DZ(4) is supported by Digital Equipment Corporation.

NAME

ec – 3Com 10 Mb/s Ethernet interface

SYNTAX

device ec0 at uba0 csr 161000 vector ecrint eccollide ecxint

DESCRIPTION

The *ec* interface provides access to a 10 Mb/s Ethernet network through a 3com controller.

The hardware has 32 kilobytes of dual-ported memory on the UNIBUS. This memory is used for internal buffering by the board, and the interface code reads the buffer contents directly through the UNIBUS.

The host's Internet address is specified at boot time with an SIOCSIFADDR ioctl. The *ec* interface employs the address resolution protocol described in *arp*(4P) to dynamically map between Internet and Ethernet addresses on the local network.

The interface software implements an exponential backoff algorithm when notified of a collision on the cable. This algorithm utilizes a 16-bit mask and the VAX-11's interval timer in calculating a series of random backoff values. The algorithm is as follows:

1. Initialize the mask to be all 1's.
2. If the mask is zero, 16 retries have been made and we give up.
3. Shift the mask left one bit and formulate a backoff by masking the interval timer with the mask (this is actually the two's complement of the value).
4. Use the value calculated in step 3 to delay before retransmitting the packet. The delay is done in a software busy loop.

The interface normally tries to use a "trailer" encapsulation to minimize copying data on input and output. This may be disabled, on a per-interface basis, by setting the IFF_NOTRAILERS flag with an SIOCSIFFLAGS ioctl.

DIAGNOSTICS

ec%d: send error. After 16 retransmissions using the exponential backoff algorithm described above, the packet was dropped.

ec%d: input error (offset=%d). The hardware indicated an error in reading a packet off the cable or an illegally sized packet. The buffer offset value is printed for debugging purposes.

ec%d: can't handle af%d. The interface was handed a message with addresses formatted in an unsuitable address family; the packet was dropped.

SEE ALSO

intro(4N), inet(4F), arp(4P)

STATUS

EC(4) currently is not supported by Digital Equipment Corporation.

NAME

en – Xerox 3 Mb/s Ethernet interface

SYNTAX

device en0 at uba0 csr 161000 vector enrint enxint encollide

DESCRIPTION

The *en* interface provides access to a 3 Mb/s Ethernet network. Due to limitations in the hardware, DMA transfers to and from the network must take place in the lower 64K bytes of the UNIBUS address space.

The network number is specified with a SIOCSIFADDR ioctl; the host's address is discovered by probing the on-board Ethernet address register. No packets will be sent or accepted until a network number is supplied.

The interface software implements an exponential backoff algorithm when notified of a collision on the cable. This algorithm utilizes a 16-bit mask and the VAX-11's interval timer in calculating a series of random backoff values. The algorithm is as follows:

1. Initialize the mask to be all 1's.
2. If the mask is zero, 16 retries have been made and we give up.
3. Shift the mask left one bit and formulate a backoff by masking the interval timer with the mask (this is actually the two's complement of the value).
4. Use the value calculated in step 3 to delay before retransmitting the packet.

The interface handles both Internet and PUP protocol families, with the interface address maintained in Internet format. PUP addresses are converted to Internet addresses by substituting PUP network and host values for Internet network and local part values.

The interface normally tries to use a "trailer" encapsulation to minimize copying data on input and output. This may be disabled, on a per-interface basis, by setting the IFF_NOTRAILERS flag with an SIOCSIFFLAGS ioctl.

DIAGNOSTICS

en%d: output error. The hardware indicated an error on the previous transmission.

en%d: send error. After 16 retransmissions using the exponential backoff algorithm described above, the packet was dropped.

en%d: input error. The hardware indicated an error in reading a packet off the cable.

en%d: can't handle af%d. The interface was handed a message with addresses formatted in an unsuitable address family; the packet was dropped.

SEE ALSO

intro(4N), inet(4F)

STATUS

EN(4) currently is not supported by Digital Equipment Corporation.

NAME

fl – console floppy interface

DESCRIPTION

This is a simple interface to the DEC RX01 floppy disk unit, which is part of the console LSI-11 subsytem for VAX-11/780's. Access is given to the entire floppy consisting of 77 tracks of 26 sectors of 128 bytes.

All i/o is raw; the seek addresses in raw transfers should be a multiple of 128 bytes and a multiple of 128 bytes should be transferred, as in other "raw" disk interfaces.

FILES

/dev/floppy

SEE ALSO

arff(8V)

DIAGNOSTICS

None.

RESTRICTIONS

Multiple console floppies are not supported.

If a write is given with a count not a multiple of 128 bytes then the trailing portion of the last sector will be zeroed.

STATUS

FL(4) is supported by Digital Equipment Corporation.

NAME

hk – RK6-11/RK06 and RK07 moving head disk

SYNTAX

controller hk0 at uba? csr 0177440 vector rkintr
disk rk0 at hk0 drive 0

DESCRIPTION

Files with minor device numbers 0 through 7 refer to various portions of drive 0; minor devices 8 through 15 refer to drive 1, etc. The standard device names begin with "hk" followed by the drive number and then a letter a-h for partitions 0-7 respectively. The character ? stands here for a drive number in the range 0-7.

The block files access the disk via the system's normal buffering mechanism and may be read and written without regard to physical disk records. There is also a 'raw' interface which provides for direct transmission between the disk and the user's read or write buffer. A single read or write call results in exactly one I/O operation and therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw files conventionally begin with an extra 'r.'

In raw I/O counts should be a multiple of 512 bytes (a disk sector). Likewise *seek* calls should specify a multiple of 512 bytes.

DISK SUPPORT

The origin and size (in sectors) of the pseudo-disks on each drive are as follows:

RK07 partitions:

disk	start	length	cyl
hk?a	0	15884	0-240
hk?b	15906	10032	241-392
hk?c	0	53790	0-814
hk?g	26004	27786	393-813

RK06 partitions

disk	start	length	cyl
hk?a	0	15884	0-240
hk?b	15906	11154	241-409
hk?c	0	27126	0-410

On a dual RK-07 system partition hk?a is used for the root for one drive and partition hk?g for the /usr file system. If large jobs are to be run using hk?b on both drives as swap area provides a 10Mbyte paging area. Otherwise partition hk?c on the other drive is used as a single large file system.

FILES

/dev/hk[0-7][a-h] block files
/dev/rhk[0-7][a-h] raw files

SEE ALSO

hp(4), uda(4), up(4), dmesg(8)

DIAGNOSTICS

The following messages are printed at the console:

rk%d%c: hard error sn%d.
An unrecoverable error occurred during transfer of the specified sector of the specified disk partition. Either the error was unrecoverable, or a large number of retry attempts (including offset positioning and drive recalibration) could not recover the error. Additional register information may be gathered from the system error log file, /usr/adm/messages.

rk%d: write locked.
The write protect switch was set on the drive when a write was attempted. The write operation is not recoverable.

rk%d: not ready.
The drive was spun down or off line when it was accessed. The i/o operation is not recoverable.

rk%d: not ready (came back!).
The drive was not ready. But after printing this message (which takes a fraction of a second), it was ready. The operation is recovered if no further errors occur.

hk%d: lost interrupt
A timer watching the controller detected no interrupt for an extended period while an operation was outstanding. This indicates a hardware or software failure. There is currently a hardware/software problem with spinning down drives while they are being accessed which causes this error to occur. The error causes a UNIBUS reset, and retry of the pending operations. If the controller continues to lose interrupts, this error will recur a few seconds later.

The following message is written to the system error log file only:

rk%d%c: soft ecc sn%d
A recoverable ECC error occurred on the specified sector in the specified disk partition. This happens normally a few times a week. If it happens more frequently than this, the sectors where the errors are occurring should be checked to see if certain cylinders on the pack, spots on the carriage of the drive or heads are indicated.

RESTRICTIONS

In raw I/O *read* and *write*(2) truncate file offsets to 512-byte block boundaries, and *write* scribbles on the tail of incomplete blocks. Thus, in programs that are likely to access raw devices, *read, write* and *lseek*(2) should always deal in 512-byte multiples.

STATUS

HK(4) is supported by Digital Equipment Corporation.

NAME

hp – MASSBUS disk interface

SYNTAX

disk hp0 at mba0 drive 0

DESCRIPTION

Files with minor device numbers 0 through 7 refer to various portions of drive 0; minor devices 8 through 15 refer to drive 1, etc. The standard device names begin with "hp" followed by the drive number and then a letter a-h for partitions 0-7 respectively. The character ? stands here for a drive number in the range 0-7.

The block file's access the disk via the system's normal buffering mechanism and may be read and written without regard to physical disk records. There is also a 'raw' interface which provides for direct transmission between the disk and the user's read or write buffer. A single read or write call results in exactly one I/O operation and therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw files conventionally begin with an extra 'r.'

In raw I/O counts should be a multiple of 512 bytes (a disk sector). Likewise *seek* calls should specify a multiple of 512 bytes.

DISK SUPPORT

This driver handles both standard DEC controllers and Emulex SC750 and SC780 controllers. Standard DEC drive types are recognized according to the MASSBUS drive type register. For the Emulex controller the drive type register should be configured to indicate the drive is an RM02. When this is encountered, the driver checks the holding register to find out the disk geometry and, based on this information, decides what the drive type is. The following disks are supported: RM03, RM05, RP06, RM80, RP05, RP07, ML11A, ML11B, CDC 9775, CDC 9730, AMPEX Capricorn (32 sectors/track), FUJITSU Eagle (48 sectors/track), and AMPEX 9300. The origin and size (in sectors) of the pseudo-disks on each drive are as follows:

RM03 partitions

disk	start	length	cyls
hp?a	0	15884	0-99
hp?b	16000	33440	100-309
hp?c	0	131680	0-822
hp?d	49600	15884	309-408
hp?e	65440	55936	409-758
hp?f	121440	10080	759-822
hp?g	49600	82080	309-822

RM05 partitions

disk	start	length	cyls
hp?a	0	15884	0-26
hp?b	16416	33440	27-81
hp?c	0	500384	0-822
hp?d	341696	15884	562-588

hp?e	358112	55936	589-680
hp?f	414048	86176	681-822
hp?g	341696	158528	562-822
hp?h	49856	291346	82-561

RP06 partitions

disk	start	length	cyls
hp?a	0	15884	0-37
hp?b	15884	33440	38-117
hp?c	0	340670	0-814
hp?d	49324	15884	118-155
hp?e	65208	55936	156-289
hp?f	121220	219296	290-814
hp?g	49324	291192	118-814

RM80 partitions

disk	start	length	cyls
hp?a	0	15884	0-36
hp?b	16058	33440	37-114
hp?c	0	242606	0-558
hp?d	49910	15884	115-151
hp?e	68096	55936	152-280
hp?f	125888	120466	281-558
hp?g	49910	192510	115-558

RP05 partitions

disk	start	length	cyls
hp?a	0	15884	0-37
hp?b	15884	33440	38-117
hp?c	0	171798	0-410
hp?d	2242	15884	118-155
hp?e	65208	55936	156-289
hp?f	121220	50424	290-410
hp?g	2242	122320	118-410

RP07 partitions

disk	start	length	cyls
hp?a	0	15884	0-9
hp?b	16000	66880	10-51
hp?c	0	1008000	0-629
hp?d	376000	15884	235-244
hp?e	392000	307200	245-436
hp?f	699200	308600	437-629
hp?g	376000	631800	235-629
hp?h	83200	291346	52-234

CDC 9775 partitions

disk	start	length	cyls
hp?a	0	15884	0-12
hp?b	16640	66880	13-65
hp?c	0	1079040	0-842
hp?d	376320	15884	294-306
hp?e	392960	307200	307-546
hp?f	700160	378720	547-842
hp?g	376320	702560	294-842
hp?h	84480	291346	66-293

CDC 9730 partitions

disk	start	length	cyls
hp?a	0	15884	0-49
hp?b	16000	33440	50-154
hp?c	0	263360	0-822
hp?d	49600	15884	155-204
hp?e	65600	55936	205-379
hp?f	121600	141600	380-822
hp?g	49600	213600	155-822

AMPEX Capricorn partitions

disk	start	length	cyls
hp?a	0	15884	0-31
hp?b	16384	33440	32-97
hp?c	0	524288	0-1023
hp?d	342016	15884	668-699
hp?e	358400	55936	700-809
hp?f	414720	109408	810-1023
hp?g	342016	182112	668-1023
hp?h	50176	291346	98-667

FUJITSU Eagle partitions

disk	start	length	cyls
hp?a	0	15884	0-16
hp?b	16320	66880	17-86
hp?c	0	808320	0-841
hp?d	375360	15884	391-407
hp?e	391680	55936	408-727
hp?f	698880	109248	728-841
hp?g	375360	432768	391-841
hp?h	83520	291346	87-390

AMPEX 9300 partitions

disk	start	length	cyl
hp?a	0	15884	0-26
hp?b	16416	33440	27-81

hp?c	0	495520	0-814
hp?d	341696	15884	562-588
hp?e	358112	55936	589-680
hp?f	414048	81312	681-814
hp?g	341696	153664	562-814
hp?h	49856	291346	82-561

It is unwise for all of these files to be present in one installation, since there is overlap in addresses and protection becomes a sticky matter. The hp?a partition is normally used for the root file system, the hp?b partition as a paging area, and the hp?c partition for pack-pack copying (it maps the entire disk). On disks larger than about 205 Megabytes, the hp?h partition is inserted prior to the hp?d or hp?g partition; the hp?g partition then maps the remainder of the pack. All disk partition tables are calculated using the *diskpart*(8) program.

FILES

/dev/hp[0-7][a-h]	block files
/dev/rhp[0-7][a-h]	raw files

SEE ALSO

hk(4), uda(4), up(4), dmesg(8)

DIAGNOSTICS

The following messages are printed at the console:

hp%d%c: hard error sn%d.
An unrecoverable error occurred during transfer of the specified sector of the named disk partition. Either the error was unrecoverable, or a large number of retry attempts (including offset positioning and drive recalibration) could not recover the error. Additional register information may be gathered from the system error log file, /usr/adm/messages.

hp%d: write locked.
The write protect switch was set on the drive when a write was attempted. The write operation is not recoverable.

hp%d: not ready.
The drive was spun down or off line when it was accessed. The i/o operation is not recoverable.

During autoconfiguration one of the following messages may appear on the console indicating the appropriate drive type was recognized. The last message indicates the drive is of a unknown type.

hp%d: 9775 (direct).
hp%d: 9730 (direct).
hp%d: 9300.
hp%d: 9762.
hp%d: capricorn.
hp%d: eagle.
hp%d: ntracks %d, nsectors %d: unknown device.

The following message is written to the system error log file only.

hp%d%c: soft ecc sn%d.
A recoverable ECC error occurred on the specified sector of the named disk partition. This happens normally a few times a week. If it happens more frequently than this, the sectors where the errors are occurring should be checked to see if certain cylinders on the pack, spots on the carriage of the drive or heads are indicated.

RESTRICTIONS

In raw I/O *read* and *write*(2) truncate file offsets to 512-byte block boundaries, and *write* scribbles on the tail of incomplete blocks. Thus, in programs that are likely to access raw devices, *read, write* and *lseek*(2) should always deal in 512-byte multiples.

STATUS

HP(4) is supported by Digital Equipment Corporation.

NAME

ht – TM-03/TE-16,TU-45,TU-77 MASSBUS magtape interface

SYNTAX

master ht0 at mba? drive ?
tape tu0 at ht0 slave 0

DESCRIPTION

The tm-03/transport combination provides a standard tape drive interface as described in *mtio*(4). All drives provide both 800 and 1600 bpi; the TE-16 runs at 45 ips, the TU-45 at 75 ips, while the TU-77 runs at 125 ips and autoloads tapes.

SEE ALSO

mt(1), tar(1), tp(1), mtio(4), tm(4), ts(4), mt(4), ut(4), dmesg(8)

DIAGNOSTICS

tu%d: no write ring.
An attempt was made to write on the tape drive when no write ring was present. This message is written on the terminal of the user who tried to access the tape.

tu%d: not online.
An attempt was made to access the tape while it was offline. This message is written on the terminal of the user who tried to access the tape.

tu%d: can't switch density in mid-tape.
An attempt was made to write on a tape at a different density than is already recorded on the tape. This message is written on the terminal of the user who tried to switch the density.

tu%d: hard error bn%d.
A tape error occurred at block *bn*. Any error is fatal on non-raw tape. When possible the driver will have retried the operation which failed several times before reporting the error. Additional register information may be gathered from the system error log file, /usr/adm/messages.

RESTRICTIONS

If any non-data error is encountered on non-raw tape, it refuses to do anything more until closed.

STATUS

HT(4) is supported by Digital Equipment Corporation.

NAME

hy – Network Systems Hyperchannel interface

SYNTAX

device hy0 at uba0 csr 0172410 vector hyint

DESCRIPTION

The *hy* interface provides access to a Network Systems Corporation Hyperchannel Adapter.

The network to which the interface is attached is specified at boot time with an SIOCSIFADDR ioctl. The host's address is discovered by reading the adapter status register. The interface will not transmit or receive packets until the network number is known.

DIAGNOSTICS

hy%d: unit number 0x%x port %d type %x microcode level 0x%x. Identifies the device during autoconfiguration.

hy%d: can't handle af%d. The interface was handed a message with addresses formatted in an unsuitable address family; the packet was dropped.

hy%d: can't initialize. The interface was unable to allocate UNIBUS resources. This is usually due to having too many network devices on an 11/750 where there are only 3 buffered data paths.

hy%d: NEX - Non Existent Memory. Non existent memory error returned from hardware.

hy%d: BAR overflow. Bus address register overflow error returned from hardware.

hy%d: Power Off bit set, trying to reset. Adapter has lost power, driver will reset the bit and see if power is still out in the adapter.

hy%d: Power Off Error, network shutdown. Power was really off in the adapter, network connections are dropped. Software does not shut down the network unless power has been off for a while.

hy%d: RECVD MP > MPSIZE (%d). A message proper was received that is too big. Probable a driver bug. Shouldn't happen.

hy%d: xmit error – len > hy_olen [%d > %d]. Probable driver error. Shouldn't happen.

hy%d: DRIVER BUG – INVALID STATE %d. The driver state machine reached a non-existent state. Definite driver bug.

hy%d: watchdog timer expired. A command in the adapter has taken too long to complete. Driver will abort and retry the command.

hy%d: adapter power restored. Software was able to reset the power off bit, indicating that the power has been restored.

SEE ALSO

intro(4N), inet(4F)

RESTRICTIONS

If the adapter does not respond to the status command issued during autoconfigure, the adapter is assumed down. A reboot is required to recognize it.

STATUS

HY(4) currently is not supported by Digital Equipment Corporation.

NAME

ik – Ikonas frame buffer, graphics device interface

SYNTAX

device ik0 at uba? csr 0172460 vector ikintr

DESCRIPTION

Ik provides an interface to an Ikonas frame buffer graphics device. Each minor device is a different frame buffer interface board. When the device is opened, its interface registers are mapped, via virtual memory, into the user processes address space. This allows the user process very high bandwidth to the frame buffer with no system call overhead.

Bytes written or read from the device are DMA'ed from or to the interface. The frame buffer XY address, its addressing mode, etc. must be set up by the user process before calling write or read.

Other communication with the driver is via ioctls. The IK_GETADDR ioctl returns the virtual address where the user process can find the interface registers. The IK_WAITINT ioctl suspends the user process until the ikonas device has interrupted (for whatever reason — the user process has to set the interrupt enables).

FILES

/dev/ik

DIAGNOSTICS

None.

RESTRICTIONS

An invalid access (e.g., longword) to a mapped interface register can cause the system to crash with a machine check.

STATUS

IK(4) currently is not supported by Digital Equipment Corporation.

NAME

il – Interlan 10 Mb/s Ethernet interface

SYNTAX

device il0 at uba0 csr 161000 vector ilrint ilcint

DESCRIPTION

The *il* interface provides access to a 10 Mb/s Ethernet network through an Interlan controller.

The host's Internet address is specified at boot time with an SIOCSIFADDR ioctl. The *il* interface employs the address resolution protocol described in *arp*(4P) to dynamically map between Internet and Ethernet addresses on the local network.

The interface normally tries to use a "trailer" encapsulation to minimize copying data on input and output. This may be disabled, on a per-interface basis, by setting the IFF_NOTRAILERS flag with an SIOCSIFFLAGS ioctl.

DIAGNOSTICS

il%d: input error. The hardware indicated an error in reading a packet off the cable or an illegally sized packet.

il%d: can't handle af%d. The interface was handed a message with addresses formatted in an unsuitable address family; the packet was dropped.

SEE ALSO

intro(4N), inet(4F), arp(4P)

STATUS

IL(4) currently is not supported by Digital Equipment Corporation.

NAME

imp – 1822 network interface

SYNTAX

pseudo-device imp

DESCRIPTION

The *imp* interface, as described in BBN Report 1822, provides access to an intelligent message processor normally used when participating in the Department of Defense ARPA network. The network interface communicates through a device controller, usually an ACC LH/DH or DEC IMP-11A, with the IMP. The interface is "reliable" and "flow-controlled" by the host-IMP protocol.

To configure IMP support, one of *acc*(4) and *css(4)* must be included. The network number on which the interface resides is specified at boot time using the SIOCSIFADDR ioctl. The host number is discovered through receipt of NOOP messages from the IMP.

The network interface is always in one of four states: up, down, initializing, or going down. When the system is booted, the interface is marked down. If the hardware controller is successfully probed, the interface enters the initializing state and transmits three NOOP messages to the IMP. It then waits for the IMP to respond with two or more NOOP messages in reply. When it receives these messages it enters the up state. The going down state is entered only when notified by the IMP of an impending shutdown. Packets may be sent through the interface only while it is in the up state. Packets received in any other state are dropped with the error ENETDOWN returned to the caller.

DIAGNOSTICS

imp%d: leader error. The IMP reported an error in a leader (1822 message header). This causes the interface to be reset and any packets queued up for transmission to be purged.

imp%d: going down in 30 seconds.
imp%d: going down for hardware PM.
imp%d: going down for reload software.
imp%d: going down for emergency reset. The Network Control Center (NCC) is manipulating the IMP. By convention these messages are reported to all hosts on an IMP.

imp%d: reset (host %d/imp %d). The host has received a NOOP message which caused it to reset its notion of its current address. This normally occurs at boot time, though it may also occur while the system is running (for example, if the IMP-controller cable is disconnected, then reconnected).

imp%d: host dead. The IMP has noted a host, to which a prior packet was sent, is not up.

imp%d: host unreachable. The IMP has discovered a host, to which a prior packet was sent, is not accessible.

imp%d: data error. The IMP noted an error in data transmitted. The host-IMP interface is reset and the host enters the init state (awaiting NOOP messages).

imp%d: interface reset. The reset process has been completed.

imp%d: marked down. After receiving a "going down in 30 seconds" message, and waiting 30 seconds, the host has marked the IMP unavailable. Before packets may be sent to the IMP again, the IMP must notify the host, through a series of NOOP messages, that it is back up.

imp%d: can't handle af%d. The interface was handed a message with addresses formatting in an unsuitable address family; the packet was dropped.

SEE ALSO

intro(4N), inet(4F), acc(4), css(4)

STATUS

IMP(4) currently is not supported by Digital Equipment Corporation.

NAME

imp – IMP raw socket interface

SYNTAX

#include <sys/socket.h>
#include <netinet/in.h>
#include <netimp/if imp.h>

s = socket(AF_IMPLINK, SOCK_RAW, IMPLINK_IP);

DESCRIPTION

The raw imp socket provides direct access to the *imp*(4) network interface. Users send packets through the interface using the *send*(2) calls, and receive packets with the *recv*(2), calls. All outgoing packets must have space for an 1822 96-bit leader on the front. Likewise, packets received by the user will have this leader on the front. The 1822 leader and the legal values for the various fields are defined in the include file *<netimp/if_imp.h>*.

The raw imp interface automatically installs the length and destination address in the 1822 leader of all outgoing packets; these need not be filled in by the user.

DIAGNOSTICS

An operation on a socket may fail with one of the following errors:

[EISCONN] when trying to establish a connection on a socket which already has one, or when trying to send a datagram with the destination address specified and the socket is already connected;

[ENOTCONN] when trying to send a datagram, but no destination address is specified, and the socket hasn't been connected;

[ENOBUFS] when the system runs out of memory for an internal data structure;

[EADDRNOTAVAIL] when an attempt is made to create a socket with a network address for which no network interface exists.

SEE ALSO

intro(4N), inet(4F), imp(4)

STATUS

IMP(4P) currently is not supported by Digital Equipment Corporation.

NAME

inet – Internet protocol family

SYNTAX

#include <sys/types.h>
#include <netinet/in.h>

DESCRIPTION

The Internet protocol family is a collection of protocols layered atop the *Internet Protocol* (IP) transport layer, and utilizing the Internet address format. The Internet family provides protocol support for the SOCK_STREAM, SOCK_DGRAM, and SOCK_RAW socket types; the SOCK_RAW interface provides access to the IP protocol.

ADDRESSING

Internet addresses are four byte quantities, stored in network standard format (on the VAX these are word and byte reversed). The include file *<netinet/in.h>* defines this address as a discriminated union.

Sockets bound to the Internet protocol family utilize the following addressing structure,

```
struct sockaddr_in {
        short       sin_family;
        u_short     sin_port;
        struct      in_addr sin_addr;
        char        sin_zero[8];
};
```

Sockets may be created with the address INADDR_ANY to effect "wildcard" matching on incoming messages.

PROTOCOLS

The Internet protocol family is comprised of the IP transport protocol, Internet Control Message Protocol (ICMP), Transmission Control Protocol (TCP), and User Datagram Protocol (UDP). TCP is used to support the SOCK_STREAM abstraction while UDP is used to support the SOCK_DGRAM abstraction. A raw interface to IP is available by creating an Internet socket of type SOCK_RAW. The ICMP message protocol is not directly accessible.

SEE ALSO

tcp(4P), udp(4P), ip(4P)

STATUS

INET(4F) currently is not supported by Digital Equipment Corporation.

NAME

ip – Internet Protocol

SYNTAX

#include <sys/socket.h>
#include <netinet/in.h>

s = socket(AF_INET, SOCK_RAW, 0);

DESCRIPTION

IP is the transport layer protocol used by the Internet protocol family. It may be accessed through a "raw socket" when developing new protocols, or special purpose applications. IP sockets are connectionless, and are normally used with the *sendto* and *recvfrom* calls, though the *connect*(2) call may also be used to fix the destination for future packets (in which case the *read*(2) or *recv*(2) and *write*(2) or *send*(2) system calls may be used).

Outgoing packets automatically have an IP header prepended to them (based on the destination address and the protocol number the socket is created with). Likewise, incoming packets have their IP header stripped before being sent to the user.

DIAGNOSTICS

A socket operation may fail with one of the following errors returned:

[EISCONN] when trying to establish a connection on a socket which already has one, or when trying to send a datagram with the destination address specified and the socket is already connected;

[ENOTCONN] when trying to send a datagram, but no destination address is specified, and the socket hasn't been connected;

[ENOBUFS] when the system runs out of memory for an internal data structure;

[EADDRNOTAVAIL] when an attempt is made to create a socket with a network address for which no network interface exists.

SEE ALSO

send(2), recv(2), intro(4N), inet(4F)

STATUS

IP(4P) currently is not supported by Digital Equipment Corporation.

NAME

kg – KL-11/DL-11W line clock

SYNTAX

device kg0 at uba0 csr 0176500 vector kglock

DESCRIPTION

A kl-11 or dl-11w can be used as an alternate real time clock source. When configured, certain system statistics and, optionally, system profiling work will be collected each time the clock interrupts. For optimum accuracy in profiling, the dl-11w should be configured to interrupt at the highest possible priority level. The *kg* device driver automatically calibrates itself to the line clock frequency.

SEE ALSO

kgmon(8), config(8)

STATUS

KG(4) currently is not supported by Digital Equipment Corporation.

NAME

lo – software loopback network interface

SYNTAX

pseudo-device loop

DESCRIPTION

The *loop* interface is a software loopback mechanism which may be used for performance analysis, software testing, and/or local communication. By default, the loopback interface is accessible at address 127.0.0.1 (non-standard); this address may be changed with the SIOCSIFADDR ioctl.

DIAGNOSTICS

lo%d: can't handle af%d. The interface was handed a message with addresses formatted in an unsuitable address family; the packet was dropped.

SEE ALSO

intro(4N), inet(4F)

STATUS

LO(4) currently is not supported by Digital Equipment Corporation.

NAME

lp – line printer

SYNTAX

device lp0 at uba0 csr 0177514 vector lpintr

DESCRIPTION

Lp provides the interface to any of the standard DEC line printers on an LP-11 parallel interface. When it is opened or closed, a suitable number of page ejects is generated. Bytes written are printed.

The unit number of the printer is specified by the minor device after removing the low 3 bits, which act as per-device parameters. Currently only the lowest of the low three bits is interpreted: if it is set, the device is treated as having a 64-character set, rather than a full 96-character set. In the resulting half-ASCII mode, lower case letters are turned into upper case and certain characters are escaped according to the following table:

{	(
}	)
`	'
|	!
~	^

The driver correctly interprets carriage returns, backspaces, tabs, and form feeds. Lines longer than the maximum page width are truncated. The default page width is 132 columns. This may be overridden by specifying, for example, "flags 256" .

FILES

/dev/lp

SEE ALSO

lpr(1)

DIAGNOSTICS

None.

STATUS

LP(4) is supported by Digital Equipment Corporation.

NAME

mem, kmem – main memory

DESCRIPTION

Mem is a special file that is an image of the main memory of the computer. It may be used, for example, to examine (and even to patch) the system.

Byte addresses in *mem* are interpreted as physical memory addresses. References to non-existent locations cause errors to be returned.

Examining and patching device registers is likely to lead to unexpected results when read-only or write-only bits are present.

The file *kmem* is the same as *mem* except that kernel virtual memory rather than physical memory is accessed.

On PDP11's, the I/O page begins at location 0160000 of *kmem* and per-process data for the current process begins at 0140000. On VAX 11/780 the I/O space begins at physical address 20000000(16); on an 11/750 I/O space addresses are of the form fxxxxx(16); on all VAX'en per-process data for the current process is at virtual 7ffff000(16).

FILES

/dev/mem
/dev/kmem

RESTRICTIONS

On PDP11's and VAX's, memory files are accessed one byte at a time, an inappropriate method for some device registers.

STATUS

MEM(4) is supported by Digital Equipment Corporation.

NAME

mt – TM78/TU-78 MASSBUS magtape interface

SYNTAX

master mt0 at mba? drive ?
tape mu0 at mt0 slave 0

DESCRIPTION

The tm78/tu-78 combination provides a standard tape drive interface as described in *mtio*(4). Only 1600 and 6250 bpi are supported; the TU-78 runs at 125 ips and autoloads tapes.

SEE ALSO

mt(1), tar(1), tp(1), mtio(4), tm(4), ts(4), ut(4), dmesg(8)

DIAGNOSTICS

mu%d: no write ring.
An attempt was made to write on the tape drive when no write ring was present. This message is written on the terminal of the user who tried to access the tape.

mu%d: not online.
An attempt was made to access the tape while it was offline. This message is written on the terminal of the user who tried to access the tape.

mu%d: can't switch density in mid-tape.
An attempt was made to write on a tape at a different density than is already recorded on the tape. This message is written on the terminal of the user who tried to switch the density.

mu%d: hard error bn%d.
A tape error occurred at block *bn*. Any error is fatal on non-raw tape. When possible the driver will have retried the operation which failed several times before reporting the error. Additional register information may be gathered from the system error log file, /usr/adm/messages.

mu%d: blank tape.
An attempt was made to read a blank tape (a tape without even end-of-file marks).

mu%d: offline.
During an i/o operation the device was set offline. If a non-raw tape was used in the access it is closed.

RESTRICTIONS

If any non-data error is encountered on non-raw tape, it refuses to do anything more until closed.

STATUS

MT(4) is supported by Digital Equipment Corporation.

NAME

mtio – UNIX magtape interface

DESCRIPTION

The files *mt0, ..., mt15* refer to the UNIX magtape drives, which may be on the MASSBUS using the TM03 formatter *ht*(4), or TM78 formatter, *mt*(4), or on the UNIBUS using either the TM11 or TS11 formatters *tm*(4), TU45 compatible formatters, *ut*(4), or *ts*(4). The following description applies to any of the transport/controller pairs. The files *mt0, ..., mt7* are 800bpi, *mt8, ..., mt15* are 1600bpi, and *mt16, ..., mt23* are 6250bpi. (But note that only 1600 bpi is available with the TS11.) The files *mt0, ..., mt3, mt8, ..., mt11*, and *mt16, ..., mt19* are rewound when closed; the others are not. When a file open for writing is closed, two end-of-files are written. If the tape is not to be rewound it is positioned with the head between the two tapemarks.

A standard tape consists of a series of 1024 byte records terminated by an end-of-file. To the extent possible, the system makes it possible, if inefficient, to treat the tape like any other file. Seeks have their usual meaning and it is possible to read or write a byte at a time. Writing in very small units is inadvisable, however, because it tends to create monstrous record gaps.

The *mt* files discussed above are useful when it is desired to access the tape in a way compatible with ordinary files. When foreign tapes are to be dealt with, and especially when long records are to be read or written, the 'raw' interface is appropriate. The associated files are named *rmt0, ..., rmt23,* but the same minor-device considerations as for the regular files still apply. A number of other ioctl operations are available on raw magnetic tape. The following definitions are from *<sys/mtio.h>*:

```
/*
 * Structures and definitions for mag tape io control commands
 */

/* structure for MTIOCTOP - mag tape op command */
struct  mtop    {
        short   mt_op;          /* operations defined below */
        daddr_t mt_count;       /* how many of them */
};

/* operations */
#define MTWEOF      0       /* write an end-of-file record */
#define MTFSF 1     /* forward space file */
#define MTBSF 2     /* backward space file */
#define MTFSR 3     /* forward space record */
#define MTBSR 4     /* backward space record */
#define MTREW       5       /* rewind */
#define MTOFFL      6       /* rewind and put the drive offline */
#define MTNOP 7     /* no operation, sets status only */
```

```
/* structure for MTIOCGET - mag tape get status command */

struct    mtget   {
          short     mt_type; /* type of magtape device */
/* the following two registers are grossly device dependent */
          short     mt_dsreg;          /* "drive status" register */
          short     mt_erreg;          /* "error" register */
/* end device-dependent registers */
          short     mt_resid;          /* residual count */
/* the following two are not yet implemented */
          daddr_t   mt_fileno;         /* file number of current position */
          daddr_t   mt_blkno;          /* block number of current position */
/* end not yet implemented */
};

/*
 * Constants for mt_type byte
 */
#define MT_ISTS                 0x01
#define MT_ISHT                 0x02
#define MT_ISTM                 0x03
#define MT_ISMT                 0x04
#define MT_ISUT                 0x05
#define MT_ISCPC        0x06
#define MT_ISAR                 0x07

/* mag tape io control commands */
#define MTIOCTOP        IOW(m, 1, struct mtop)          /* do a mag tape op */
#define MTIOCGET        IOR(m, 2, struct mtget)  /* get tape status */

#ifndef KERNEL
#define DEFTAPE         "/dev/rmt12"
#endif
```

Each *read* or *write* call reads or writes the next record on the tape. In the write case the record has the same length as the buffer given. During a read, the record size is passed back as the number of bytes read, provided it is no greater than the buffer size; if the record is long, an error is indicated. In raw tape I/O seeks are ignored. A zero byte count is returned when a tape mark is read, but another read will fetch the first record of the new tape file.

FILES

/dev/mt?
/dev/rmt?

SEE ALSO

mt(1), tar(1), tp(1), ht(4), tm(4), ts(4), mt(4), ut(4)

STATUS

MTIO(4) currently is not supported by Digital Equipment Corporation.

NAME

null – data sink

DESCRIPTION

Data written on a null special file is discarded.

Reads from a null special file always return 0 bytes.

FILES

/dev/null

STATUS

NULL(4) is supported by Digital Equipment Corporation.

NAME

pcl – DEC CSS PCL-11 B Network Interface

SYNTAX

device pcl0 at uba? csr 164200 vector pclxint pclrint

DESCRIPTION

The *pcl* device provides an IP-only interface to the DEC CSS PCL-11 time division multiplexed network bus. The controller itself is not accessible to users.

The hosts's address is specified with the SIOCSIFADDR ioctl. The interface will not transmit or receive any data before its address is defined.

As the PCL-11 hardware is only capable of having 15 interfaces per network, a single-byte host-on-network number is used, with range [1..15] to match the TDM bus addresses of the interfaces.

The interface currently only supports the Internet protocol family and only provides "natural" (header) encapsulation.

DIAGNOSTICS

pcl%d: can't init. Insufficient UNIBUS resources existed to initialize the device. This is likely to occur when the device is run on a buffered data path on an 11/750 and other network interfaces are also configured to use buffered data paths, or when it is configured to use buffered data paths on an 11/730 (which has none).

pcl%d: can't handle af%d. The interface was handed a message with addresses formatted in an unsuitable address family; the packet was dropped.

pcl%d: stray xmit interrupt. An interrupt occured when no output had previously been started.

pcl%d: master. The TDM bus had no station providing "bus master" timing signals, so this interface has assumed the "master" role. This message should only appear at most once per UNIBUS INIT on a single system. Unless there is a hardware failure, only one station may be master at at time.

pcl%d: send error, tcr=%b, tsr=%b. The device indicated a problem sending data on output. If a "receiver offline" error is detected, it is not normally logged unless the option PCL_TESTING has been selected, as this causes a lot of console chatter when sending to a down machine. However, this option is quite useful when debugging problems with the PCL interfaces.

pcl%d: rcv error, rcr=%b rsr=%b. The device indicated a problem receiving data on input.

pcl%d: bad len=%d. An input operation resulted in a data transfer of less than 0 or more than 1008 bytes of data into memory (according to the word count register). This should never happen as the maximum size of a PCL message has been agreed upon to be 1008 bytes (same as ArpaNet message).

SEE ALSO

intro(4N), inet(4F)

STATUS

PCL(4) currently is not supported by Digital Equipment Corporation.

NAME

ps – Evans and Sutherland Picture System 2 graphics device interface

SYNTAX

device ps0 at uba? csr 0172460 vector psintr

DESCRIPTION

The *ps* driver provides access to an Evans and Sutherland Picture System 2 graphics device. Each minor device is a new PS2. When the device is opened, its interface registers are mapped, via virtual memory, into a user process's address space. This allows the user process very high bandwidth to the device with no system call overhead.

DMA to and from the PS2 is not supported. All read and write system calls will fail. All data is moved to and from the PS2 via programmed I/O using the device's interface registers.

Commands are fed to and from the driver using the following ioctls:

PSIOGETADDR
: Returns the virtual address through which the user process can access the device's interface registers.

PSIOAUTOREFRESH
: Start auto refreshing the screen. The argument is an address in user space where the following data resides. The first longword is a *count* of the number of static refresh buffers. The next *count* longwords are the addresses in refresh memory where the refresh buffers lie. The driver will cycle thru these refresh buffers displaying them one by one on the screen.

PSIOAUTOMAP
: Start automatically passing the display file thru the matrix processor and into the refresh buffer. The argument is an address in user memory where the following data resides. The first longword is a *count* of the number of display files to operate on. The next *count* longwords are the address of these display files. The final longword is the address in refresh buffer memory where transformed coordinates are to be placed if the driver is not in double buffer mode (see below).

PSIODOUBLEBUFFER
: Cause the driver to double buffer the output from the map that is going to the refresh buffer. The argument is again a user space address where the real arguments are stored. The first argument is the starting address of refresh memory where the two double buffers are located. The second argument is the length of each double buffer. The refresh mechanism displays the current double buffer, in addition to its static refresh lists, when in double buffer mode.

PSIOSINGLEREFRESH
: Single step the refresh process. That is, the driver does not continually refresh the screen.

PSIOSINGLEMAP

Single step the matrix process. The driver does not automatically feed display files thru the matrix unit.

PSIOSINGLEBUFFER
Turn off double buffering.

PSIOTIMEREFRESH
The argument is a count of the number of refresh interrupts to take before turning off the screen. This is used to do time exposures.

PSIOWAITREFRESH
Suspend the user process until a refresh interrupt has occurred. If in TIMEREFRESH mode, suspend until count refreshes have occurred.

PSIOSTOPREFRESH
Wait for the next refresh, stop all refreshes, and then return to user process.

PSIOWAITMAP
Wait until a map done interrupt has occurred.

PSIOSTOPMAP
Wait for a map done interrupt, do not restart the map, and then return to the user.

FILES

/dev/ps

DIAGNOSTICS

ps device intr.

ps dma intr. An interrupt was received from the device. This shouldn't happen, check your device configuration for overlapping interrupt vectors.

RESTRICTIONS

An invalid access (e.g., longword) to a mapped interface register can cause the system to crash with a machine check.

STATUS

PS(4) currently is not supported by Digital Equipment Corporation.

NAME

pty – pseudo terminal driver

SYNTAX

pseudo-device pty

DESCRIPTION

The *pty* driver provides support for a device-pair termed a *pseudo terminal*. A pseudo terminal is a pair of character devices, a *master* device and a *slave* device. The slave device provides processes an interface identical to that described in *tty*(4). However, whereas all other devices which provide the interface described in *tty*(4) have a hardware device of some sort behind them, the slave device has, instead, another process manipulating it through the master half of the pseudo terminal. That is, anything written on the master device is given to the slave device as input and anything written on the slave device is presented as input on the master device.

In configuring, if no optional "count" is given in the specification, 16 pseudo terminal pairs are configured.

The following *ioctl* calls apply only to pseudo terminals:

TIOCSTOP
: Stops output to a terminal (e.g. like typing ^S). Takes no parameter.

TIOCSTART
: Restarts output (stopped by TIOCSTOP or by typing ^S). Takes no parameter.

TIOCPKT
: Enable/disable *packet* mode. Packet mode is enabled by specifying (by reference) a nonzero parameter and disabled by specifying (by reference) a zero parameter. When applied to the master side of a pseudo terminal, each subsequent *read* from the terminal will return data written on the slave part of the pseudo terminal preceded by a zero byte (symbolically defined as TIOCPKT_DATA), or a single byte reflecting control status information. In the latter case, the byte is an inclusive-or of zero or more of the bits:

  TIOCPKT_FLUSHREAD
  : whenever the read queue for the terminal is flushed.

  TIOCPKT_FLUSHWRITE
  : whenever the write queue for the terminal is flushed.

  TIOCPKT_STOP
  : whenever output to the terminal is stopped a la ^S.

  TIOCPKT_START
  : whenever output to the terminal is restarted.

  TIOCPKT_DOSTOP
  : whenever *t stopc* is ^S and *t startc* is ^Q.

  TIOCPKT_NOSTOP

whenever the start and stop characters are not ^S/^Q.

This mode is used by *rlogin*(1C) and *rlogind*(8C) to implement a remote-echoed, locally ^S/^Q flow-controlled remote login with proper back-flushing of output; it can be used by other similar programs.

TIOCREMOTE

A mode for the master half of a pseudo terminal, independent of TIOCPKT. This mode causes input to the pseudo terminal to be flow controlled and not input edited (regardless of the terminal mode). Each write to the control terminal produces a record boundary for the process reading the terminal. In normal usage, a write of data is like the data typed as a line on the terminal; a write of 0 bytes is like typing an end-of-file character. TIOCREMOTE can be used when doing remote line editing in a window manager, or whenever flow controlled input is required.

FILES

/dev/pty[p-r][0-9a-f]	master pseudo terminals
/dev/tty[p-r][0-9a-f]	slave pseudo terminals

DIAGNOSTICS

None.

RESTRICTIONS

It is not possible to send an EOT.

STATUS

PTY(4) currently is not supported by Digital Equipment Corporation.

NAME

pup – Xerox PUP-I protocol family

SYNTAX

#include <sys/types.h>
#include <netpup/pup.h>

DESCRIPTION

The PUP-I protocol family is a collection of protocols layered atop the PUP Level-0 packet format, and utilizing the PUP Internet address format. The PUP family is currently supported only by a raw interface.

ADDRESSING

PUP addresses are composed of network, host, and port portions. The include file *<netpup/pup.h>* defines this address as,

```
struct   pupport {
         u_char      pup_net;
         u_char      pup_host;
         u_char      pup_socket[4];
};
```

Sockets bound to the PUP protocol family utilize the following addressing structure,

```
struct sockaddr_pup {
         short       spup_family;
         short       spup_zero1;
         u_char      spup_net;
         u_char      spup_host;
         u_char      spup_sock[4];
         char        spup_zero2[4];
};
```

HEADERS

The current PUP support provides only raw access to the 3Mb/s Ethernet. Packets sent through this interface must have space for the following packet header present at the front of the message,

```
struct pup header {
         u_short     pup_length;
         u_char      pup_tcontrol;      /* transport control */
         u_char      pup_type;          /* protocol type */
         u_long      pup_id;            /* used by protocols */
         u_char      pup_dnet;          /* destination */
         u_char      pup_dhost;
         u_char      pup_dsock[4];
         u_char      pup_snet;          /* source */
         u_char      pup_shost;
         u_char      pup_ssock[4];
```

};

The sender should fill in the *pup_tcontrol*, *pup_type*, and *pup_id* fields. The remaining fields are filled in by the system. The system checks the message to insure its size is valid and, calulates a checksum for the message. If no checksum should be calculated, the checksum field (the last 16-bit word in the message) should be set to PUP NOCKSUM.

The *pup_tcontrol* field is restricted to be 0 or PUP_TRACE; PUP_TRACE indicates packet tracing should be performed. The *pup_type* field may not be 0.

On input, the entire packet, including header, is provided the user. No checksum validation is performed.

SEE ALSO

intro(4N), pup(4P), en(4)

STATUS

PUP(4F) currently is not supported by Digital Equipment Corporation.

NAME

pup – raw PUP socket interface

SYNTAX

#include <sys/socket.h>
#include <netpup/pup.h>

socket(AF_PUP, SOCK_RAW, PUPPROTO_BSP);

DESCRIPTION

A raw pup socket provides PUP-I access to an Ethernet network. Users send packets using the *sendto* call, and receive packets with the *recvfrom* call. All outgoing packets must have space present at the front of the packet to allow the PUP header to be filled in. The header format is described in *pup*(4F). Likewise, packets received by the user will have the PUP header on the front. The PUP header and legal values for the various fields are defined in the include file *<netpup/pup.h>*.

The raw pup interface automatically installs the length and source and destination addresses in the PUP header of all outgoing packets; these need not be filled in by the user. The only control bit that may be set in the *tcontrol* field of outgoing packets is the "trace" bit. A checksum is calculated unless the sender sets the checksum field to PUP_NOCKSUM.

DIAGNOSTICS

A socket operation may fail and one of the following will be returned:

[EISCONN] when trying to establish a connection on a socket which already has one, or when trying to send a datagram with the destination address specified and the socket is already connected;

[ENOTCONN] when trying to send a datagram, but no destination address is specified, and the socket hasn't been connected;

[ENOBUFS] when the system runs out of memory for an internal data structure;

[EADDRNOTAVAIL] when an attempt is made to create a socket with a network address for which no network interface exists.

A *sendto* operation may fail if one of the following is true:

[EINVAL] Insufficient space was left by the user for the PUP header.

[EINVAL] The *pup_type* field was 0 or the *pup_tcontrol* field had a bit other than PUP_TRACE set.

[EMSGSIZE] The message was not an even number of bytes, smaller than MINPUPSIZ, or large than MAXPUPSIZ.

[ENETUNREACH] The destination address was on a network which was not directly reachable (the raw interface provides no routing support).

SEE ALSO

send(2), recv(2), intro(4N), pup(4F)

STATUS

PUP(4P) currently is not supported by Digital Equipment Corporation.

NAME

rx – DEC RX02 floppy disk interface

SYNTAX

controller fx0 at uba0 csr 0177170 vector rxintr
disk rx0 at fx0 slave 0
disk rx1 at fx0 slave 1

DESCRIPTION

The *rx* device provides access to a DEC RX02 floppy disk unit with M8256 interface module (RX211 configuration). The RX02 uses 8-inch, single-sided, soft-sectored floppy disks (with pre-formatted industry-standard headers) in either single or double density.

Floppy disks handled by the RX02 contain 77 tracks, each with 26 sectors (for a total of 2,002 sectors). The sector size is 128 bytes for single density, 256 bytes for double density. Single density disks are compatible with the RX01 floppy disk unit and with IBM 3740 Series Diskette 1 systems.

In addition to normal ('block' and 'raw') i/o, the driver supports formatting of disks for either density and the ability to invoke a 2 for 1 interleaved sector mapping compatible with the DEC operating system RT-11.

The minor device number is interpreted as follows:

Bit	Description
0	Sector interleaving (1 disables interleaving)
1	Logical sector 1 is on track 1 (0 no, 1 yes)
2	Not used, reserved
Other	Drive number

The two drives in a single RX02 unit are treated as two disks attached to a single controller. Thus, if there are two RX02's on a system, the drives on the first RX02 are "rx0" and "rx1", while the drives on the second are "rx2" and "rx3".

When the device is opened, the density of the disk currently in the drive is automatically determined. If there is no floppy in the device, open will fail.

The interleaving parameters are represented in raw device names by the letters 'a' through 'd'. Thus, unit 0, drive 0 is called by one of the following names:

Mapping	Device name	Starting track
interleaved	/dev/rrx0a	0
direct	/dev/rrx0b	0
interleaved	/dev/rrx0c	1
direct	/dev/rrx0d	1

The mapping used on the 'c' device is compatible with the DEC operating system RT-11. The 'b' device accesses the sectors of the disk in strictly sequential order. The 'a' device is the most efficient for disk-to-disk copying.

I/O requests must start on a sector boundary, involve an integral number of complete sectors, and not go off the end of the disk.

NOTES

Even though the storage capacity on a floppy disk is quite small, it is possible to make filesystems on double density disks. For example, the command

```
% mkfs /dev/rx0 1001 13 1 4096 512 32 0 4
```

makes a file system on the double density disk in rx0 with 436 kbytes available for file storage. Using *tar*(1) gives a more efficient utilization of the available space for file storage. Single density diskettes do not provide sufficient storage capacity to hold file systems.

A number of *ioctl*(2) calls apply to the rx devices, and have the form

```
#include <vaxuba/rxreg.h>
ioctl(fildes, code, arg)
int *arg;
```

The applicable codes are:

RXIOC_FORMAT
: Format the diskette. The density to use is specified by the *arg* argument, 0 gives single density while non-zero gives double density.

RXIOC_GETDENS
: Return the density of the diskette (0 or !=0 as above).

RXIOC_WDDMK
: On the next write, include a *deleted data address mark* in the header of the first sector.

RXIOC_RDDMK
: Return non-zero if the last sector read contained a *deleted data address mark* in its header, otherwise return 0.

ERRORS

The following errors may be returned by the above ioctl calls:

[ENODEV] Drive not ready; usually because no disk is in the drive or the drive door is open.

[ENXIO] Nonexistent drive (on open); offset is too large or not on a sector boundary or byte count is not a multiple of the sector size (on read or write); or bad (undefined) ioctl code.

[EIO] A physical error other than "not ready", probably bad media or unknown format.

[EBUSY] Drive has been opened for exclusive access.

[EBADF] No write access (on format), or wrong density; the latter can only happen if the disk is changed without closing the device (i.e., calling *close*(2)).

FILES

/dev/rx?
/dev/rrx?[a-d]

SEE ALSO

rxformat(8V), newfs(8), mkfs(8), tar(1), arff(8V)

DIAGNOSTICS

rx%d: hard error, trk %d psec %d cs=%b, db=%b, err=%x, %x, %x, %x. An unrecoverable error was encountered. The track and physical sector numbers, the device registers and the extended error status are displayed.

rx%d: state %d (reset). The driver entered a bogus state. This should not happen.

RESTRICTIONS

A floppy may not be formatted if the header information on sector 1, track 0 has been damaged. Hence, it is not possible to format completely degaussed disks or disks with other formats than the two known by the hardware.

If the drive subsystem is powered down when the machine is booted, the controller won't interrupt.

STATUS

RX(4) currently is not supported by Digital Equipment Corporation.

NAME

tcp – Internet Transmission Control Protocol

SYNTAX

#include <sys/socket.h>
#include <netinet/in.h>

s = socket(AF_INET, SOCK_STREAM, 0);

DESCRIPTION

The TCP protocol provides reliable, flow-controlled, two-way transmission of data. It is a byte-stream protocol used to support the SOCK STREAM abstraction. TCP uses the standard Internet address format and, in addition, provides a per-host collection of "port addresses". Thus, each address is composed of an Internet address specifying the host and network, with a specific TCP port on the host identifying the peer entity.

Sockets utilizing the tcp protocol are either "active" or "passive". Active sockets initiate connections to passive sockets. By default TCP sockets are created active; to create a passive socket the *listen*(2) system call must be used after binding the socket with the *bind*(2) system call. Only passive sockets may use the *accept*(2) call to accept incoming connections. Only active sockets may use the *connect*(2) call to initiate connections.

Passive sockets may "underspecify" their location to match incoming connection requests from multiple networks. This technique, termed "wildcard addressing", allows a single server to provide service to clients on multiple networks. To create a socket which listens on all networks, the Internet address INADDR_ANY must be bound. The TCP port may still be specified at this time; if the port is not specified the system will assign one. Once a connection has been established the socket's address is fixed by the peer entity's location. The address assigned the socket is the address associated with the network interface through which packets are being transmitted and received. Normally this address corresponds to the peer entity's network.

DIAGNOSTICS

A socket operation may fail with one of the following errors returned:

[EISCONN]	when trying to establish a connection on a socket which already has one;
[ENOBUFS]	when the system runs out of memory for an internal data structure;
[ETIMEDOUT]	when a connection was dropped due to excessive retransmissions;
[ECONNRESET]	when the remote peer forces the connection to be closed;
[ECONNREFUSED]	when the remote peer actively refuses connection establishment (usually because no process is listening to the port);
[EADDRINUSE]	when an attempt is made to create a socket with a port which has already been allocated;
[EADDRNOTAVAIL]	

when an attempt is made to create a socket with a network address for which no network interface exists.

SEE ALSO

intro(4N), inet(4F)

STATUS

TCP(4P) currently is not supported by Digital Equipment Corporation.

NAME

tm – TM-11/TE-10 magtape interface

SYNTAX

controller tm0 at uba? csr 0172520 vector tmintr
tape te0 at tm0 drive 0

DESCRIPTION

The tm-11/te-10 combination provides a standard tape drive interface as described in *mtio*(4). Hardware implementing this on the VAX is typified by the Emulex TC-11 controller operating with a Kennedy model 9300 tape transport, providing 800 and 1600 bpi operation at 125 ips.

SEE ALSO

mt(1), tar(1), tp(1), mtio(4), ht(4), ts(4), mt(4), ut(4)

DIAGNOSTICS

te%d: no write ring. An attempt was made to write on the tape drive when no write ring was present; this message is written on the terminal of the user who tried to access the tape.

te%d: not online. An attempt was made to access the tape while it was offline; this message is written on the terminal of the user who tried to access the tape.

te%d: can't switch density in mid-tape. An attempt was made to write on a tape at a different density than is already recorded on the tape. This message is written on the terminal of the user who tried to switch the density.

te%d: hard error bn%d er=%b. A tape error occurred at block *bn*; the tm error register is printed in octal with the bits symbolically decoded. Any error is fatal on non-raw tape; when possible the driver will have retried the operation which failed several times before reporting the error.

te%d: lost interrupt. A tape operation did not complete within a reasonable time, most likely because the tape was taken off-line during rewind or lost vacuum. The controller should, but does not, give an interrupt in these cases. The device will be made available again after this message, but any current open reference to the device will return an error as the operation in progress aborts.

RESTRICTIONS

If any non-data error is encountered on non-raw tape, it refuses to do anything more until closed.

STATUS

TM(4) currently is not supported by Digital Equipment Corporation.

NAME

ts – TS-11 magtape interface

SYNTAX

controller zs0 at uba? csr 0172520 vector tsintr
tape ts0 at zs0 drive 0

DESCRIPTION

The ts-11 combination provides a standard tape drive interface as described in *mtio*(4). The ts-11 operates only at 1600 bpi, and only one transport is possible per controller.

SEE ALSO

mt(1), tar(1), tp(1), mtio(4), ht(4), tm(4), mt(4), ut(4), dmesg(8)

DIAGNOSTICS

ts%d: no write ring.
An attempt was made to write on the tape drive when no write ring was present. This message is written on the terminal of the user who tried to access the tape.

ts%d: not online.
An attempt was made to access the tape while it was offline. This message is written on the terminal of the user who tried to access the tape.

ts%d: hard error bn%d.
A hard error occurred on the tape at block *bn*. Additional register information may be gathered from the system error log file, /usr/adm/messages.

RESTRICTIONS

If any non-data error is encountered on non-raw tape, it refuses to do anything more until closed.

The device lives at the same address as a tm-11 *tm*(4); as it is very difficult to get this device to interrupt, a generic system assumes that a ts is present whenever no tm-11 exists but the csr responds and a ts-11 is configured. This does no harm as long as a non-existent ts-11 is not accessed.

STATUS

TS(4) is supported by Digital Equipment Corporation.

NAME

tty – general terminal interface

SYNTAX

#include <sgtty.h>

DESCRIPTION

This section describes both a particular special file **/dev/tty** and the terminal drivers used for conversational computing.

Line disciplines.

The system provides different *line disciplines* for controlling communications lines. In this version of the system there are three disciplines available:

old The old (standard) terminal driver. This is used when using the standard shell *sh*(1) and for compatibility with other standard version 7 UNIX systems.

new A newer terminal driver, with features for job control; this must be used when using *csh*(1).

net A line discipline used for networking and loading data into the system over communications lines. It allows high speed input at very low overhead, and is described in *bk*(4).

Line discipline switching is accomplished with the TIOCSETD *ioctl:*

int ldisc = LDISC; ioctl(filedes, TIOCSETD, &ldisc);

where LDISC is OTTYDISC for the standard tty driver, NTTYDISC for the new driver and NETLDISC for the networking discipline. The standard (currently old) tty driver is discipline 0 by convention. The current line discipline can be obtained with the TIOCGETD ioctl. Pending input is discarded when the line discipline is changed.

All of the low-speed asynchronous communications ports can use any of the available line disciplines, no matter what hardware is involved. The remainder of this section discusses the "old" and "new" disciplines.

The control terminal.

When a terminal file is opened, it causes the process to wait until a connection is established. In practice, user programs seldom open these files; they are opened by *init*(8) and become a user's standard input and output file.

If a process which has no control terminal opens a terminal file, then that terminal file becomes the control terminal for that process. The control terminal is thereafter inherited by a child process during a *fork*(2), even if the control terminal is closed.

The file **/dev/tty** is, in each process, a synonym for a *control terminal* associated with that process. It is useful for programs that wish to be sure of writing messages on the terminal no matter how output has been redirected. It can also be used for programs that demand a file name for output, when typed output is desired and it is tiresome to find out which terminal is currently in use.

Process groups.

Command processors such as *csh*(1) can arbitrate the terminal between different *jobs* by placing related jobs in a single process group and associating this process group with the terminal. A terminals associated process group may be set using the TIOCSPGRP *ioctl*(2):

ioctl(fildes, TIOCSPGRP, &pgrp)

or examined using TIOCGPGRP rather than TIOCSPGRP, returning the current process group in *pgrp*. The new terminal driver aids in this arbitration by restricting access to the terminal by processes which are not in the current process group; see **Job access control** below.

Modes.

The terminal drivers have three major modes, characterized by the amount of processing on the input and output characters:

cooked The normal mode. In this mode lines of input are collected and input editing is done. The edited line is made available when it is completed by a newline or when an EOT (control-D, hereafter ^D) is entered. A carriage return is usually made synonymous with newline in this mode, and replaced with a newline whenever it is typed. All driver functions (input editing, interrupt generation, output processing such as delay generation and tab expansion, etc.) are available in this mode.

CBREAK
This mode eliminates the character, word, and line editing input facilities, making the input character available to the user program as it is typed. Flow control, literal-next and interrupt processing are still done in this mode. Output processing is done.

RAW This mode eliminates all input processing and makes all input characters available as they are typed; no output processing is done either.

The style of input processing can also be very different when the terminal is put in non-blocking i/o mode; see *fcntl*(2). In this case a *read*(2) from the control terminal will never block, but rather return an error indication (EWOULDBLOCK) if there is no input available.

A process may also request a SIGIO signal be sent it whenever input is present. To enable this mode the FASYNC flag should be set using *fcntl*(2).

Input editing.

A UNIX terminal ordinarily operates in full-duplex mode. Characters may be typed at any time, even while output is occurring, and are only lost when the system's character input buffers become completely choked, which is rare, or when the user has accumulated the maximum allowed number of input characters that have not yet been read by some program. Currently this limit is 256 characters. In the old terminal driver all the saved characters are thrown away when the limit is reached, without notice; the new driver simply refuses to accept any further input, and rings the terminal bell.

Input characters are normally accepted in either even or odd parity with the parity bit being stripped off before the character is given to the program. By clearing either the EVEN or ODD bit in the flags word it is possible to have input characters with that parity discarded (see the **Summary** below.)

In all of the line disciplines, it is possible to simulate terminal input using the TIOCSTI ioctl, which takes, as its third argument, the address of a character. The system pretends that this character was typed on the argument terminal, which must be the control terminal except for the super-user (this call is not in standard version 7 UNIX).

Input characters are normally echoed by putting them in an output queue as they arrive. This may be disabled by clearing the ECHO bit in the flags word using the *stty*(3) call or the TIOCSETN or TIOCSETP ioctls (see the **Summary** below).

In cooked mode, terminal input is processed in units of lines. A program attempting to read will normally be suspended until an entire line has been received (but see the description of SIGTTIN in **Modes** above and FIONREAD in **Summary** below.) No matter how many characters are requested in the read call, at most one line will be returned. It is not, however, necessary to read a whole line at once; any number of characters may be requested in a read, even one, without losing information.

During input, line editing is normally done, with the character '#' logically erasing the last character typed and the character '@' logically erasing the entire current input line. These are often reset on crt's, with ^H replacing #, and ^U replacing @. These characters never erase beyond the beginning of the current input line or an ^D. These characters may be entered literally by preceding them with '\'; in the old teletype driver both the '\' and the character entered literally will appear on the screen; in the new driver the '\' will normally disappear.

The drivers normally treat either a carriage return or a newline character as terminating an input line, replacing the return with a newline and echoing a return and a line feed. If the CRMOD bit is cleared in the local mode word then the processing for carriage return is disabled, and it is simply echoed as a return, and does not terminate cooked mode input.

In the new driver there is a literal-next character ^V which can be typed in both cooked and CBREAK mode preceding **any** character to prevent its special meaning. This is to be preferred to the use of '\' escaping erase and kill characters, but '\' is (at least temporarily) retained with its old function in the new driver for historical reasons.

The new terminal driver also provides two other editing characters in normal mode. The word-erase character, normally ^W, erases the preceding word, but not any spaces before it. For the purposes of ^W, a word is defined as a sequence of non-blank characters, with tabs counted as blanks. Finally, the reprint character, normally ^R, retypes the pending input beginning on a new line. Retyping occurs automatically in cooked mode if characters which would normally be erased from the screen are fouled by program output.

Input echoing and redisplay

In the old terminal driver, nothing special occurs when an erase character is typed; the erase character is simply echoed. When a kill character is typed it is echoed followed by a new-line (even if the character is not killing the line, because it was preceded by a '\'!.)

The new terminal driver has several modes for handling the echoing of terminal input, controlled by bits in a local mode word.

Hardcopy terminals. When a hardcopy terminal is in use, the LPRTERA bit is normally set in the local mode word. Characters which are logically erased are then printed out backwards preceded by '\' and followed by '/' in this mode.

Crt terminals. When a crt terminal is in use, the LCRTBS bit is normally set in the local mode word. The terminal driver then echoes the proper number of erase characters when input is erased; in the normal case where the erase character is a ^H this causes the cursor of the terminal to back up to where it was before the logically erased character was typed. If the input has become fouled due to interspersed asynchronous output, the input is automatically retyped.

Erasing characters from a crt. When a crt terminal is in use, the LCRTERA bit may be set to cause input to be erased from the screen with a "backspace-space-backspace" sequence when character or word deleting sequences are used. A LCRTKIL bit may be set as well, causing the input to be erased in this manner on line kill sequences as well.

Echoing of control characters. If the LCTLECH bit is set in the local state word, then non-printing (control) characters are normally echoed as ^X (for some X) rather than being echoed unmodified; delete is echoed as ^?.

The normal modes for using the new terminal driver on crt terminals are speed dependent. At speeds less than 1200 baud, the LCRTERA and LCRTKILL processing is painfully slow, so *stty*(1) normally just sets LCRTBS and LCTLECH; at speeds of 1200 baud or greater all of these bits are normally set. *Stty*(1) summarizes these option settings and the use of the new terminal driver as "newcrt."

Output processing.

When one or more characters are written, they are actually transmitted to the terminal as soon as previously-written characters have finished typing. (As noted above, input characters are normally echoed by putting them in the output queue as they arrive.) When a process produces characters more rapidly than they can be typed, it will be suspended when its output queue exceeds some limit. When the queue has drained down to some threshold the program is resumed. Even parity is normally generated on output. The EOT character is not transmitted in cooked mode to prevent terminals that respond to it from hanging up; programs using raw or cbreak mode should be careful.

The terminal drivers provide necessary processing for cooked and CBREAK mode output including delay generation for certain special characters and parity generation. Delays are available after backspaces ^H, form feeds ^L, carriage returns ^M, tabs ^I and newlines ^J. The driver will also optionally expand tabs into spaces, where the tab stops are assumed to be set every eight columns. These functions are controlled by bits in the tty flags word; see **Summary** below.

The terminal drivers provide for mapping between upper and lower case on terminals lacking lower case, and for other special processing on deficient terminals.

Finally, in the new terminal driver, there is a output flush character, normally ^O, which sets the LFLUSHO bit in the local mode word, causing subsequent output to be flushed until it is cleared by a program or more input is typed. This character has effect in both cooked and CBREAK modes and causes pending input to be retyped if there is any pending input. An ioctl to flush the characters in the input and output queues TIOCFLUSH, is also available.

Upper case terminals and Hazeltines

If the LCASE bit is set in the tty flags, then all upper-case letters are mapped into the corresponding lower-case letter. The upper-case letter may be generated by preceding it by '\'. If the new terminal driver is being used, then upper case letters are preceded by a '\' when output. In addition, the following escape sequences can be generated on output and accepted on input:

for	`	|	~	{	}
use	\'	\!	\^	\(	\)

To deal with Hazeltine terminals, which do not understand that ~ has been made into an ASCII character, the LTILDE bit may be set in the local mode word when using the new terminal driver; in this case the character ~ will be replaced with the character ` on output.

Flow control.

There are two characters (the stop character, normally ^S, and the start character, normally ^Q) which cause output to be suspended and resumed respectively. Extra stop characters typed when output is already stopped have no effect, unless the start and stop characters are made the same, in which case output resumes.

A bit in the flags word may be set to put the terminal into TANDEM mode. In this mode the system produces a stop character (default ^S) when the input queue is in danger of overflowing, and a start character (default ^Q) when the input has drained sufficiently. This mode is useful when the terminal is actually another machine that obeys the conventions.

Line control and breaks.

There are several *ioctl* calls available to control the state of the terminal line. The TIOCSBRK ioctl will set the break bit in the hardware interface causing a break condition to exist; this can be cleared (usually after a delay with *sleep*(3)) by TIOCCBRK. Break conditions in the input are reflected as a null character in RAW mode or as the interrupt character in cooked or CBREAK mode. The TIOCCDTR ioctl will clear the data terminal ready condition; it can be set again by TIOCSDTR.

When the carrier signal from the dataset drops (usually because the user has hung up his terminal) a SIGHUP hangup signal is sent to the processes in the distinguished process group of the terminal; this usually causes them to terminate (the SIGHUP can be suppressed by setting the LNOHANG bit in the local state word of the driver.) Access to

the terminal by other processes is then normally revoked, so any further reads will fail, and programs that read a terminal and test for end-of-file on their input will terminate appropriately.

When using an ACU it is possible to ask that the phone line be hung up on the last close with the TIOCHPCL ioctl; this is normally done on the outgoing line.

Interrupt characters.

There are several characters that generate interrupts in cooked and CBREAK mode; all are sent the processes in the control group of the terminal, as if a TIOCGPGRP ioctl were done to get the process group and then a *killpg*(2) system call were done, except that these characters also flush pending input and output when typed at a terminal (*à"la* TIOCFLUSH). The characters shown here are the defaults; the field names in the structures (given below) are also shown. The characters may be changed, although this is not often done.

^? **t_intrc** (Delete) generates a SIGINT signal. This is the normal way to stop a process which is no longer interesting, or to regain control in an interactive program.

^\ **t_quitc** (FS) generates a SIGQUIT signal. This is used to cause a program to terminate and produce a core image, if possible, in the file **core** in the current directory.

^Z **t_suspc** (EM) generates a SIGTSTP signal, which is used to suspend the current process group.

^Y **t_dsuspc** (SUB) generates a SIGTSTP signal as ^Z does, but the signal is sent when a program attempts to read the ^Y, rather than when it is typed.

Job access control.

When using the new terminal driver, if a process which is not in the distinguished process group of its control terminal attempts to read from that terminal its process group is sent a SIGTTIN signal. This signal normally causes the members of that process group to stop. If, however, the process is ignoring SIGTTIN, has SIGTTIN blocked, is an *orphan process*, or is in the middle of process creation using *vfork*(2)), it is instead returned an end-of-file. (An *orphan process* is a process whose parent has exited and has been inherited by the *init*(8) process.) Under older UNIX systems these processes would typically have had their input files reset to **/dev/null,** so this is a compatible change.

When using the new terminal driver with the LTOSTOP bit set in the local modes, a process is prohibited from writing on its control terminal if it is not in the distinguished process group for that terminal. Processes which are holding or ignoring SIGTTOU signals, which are orphans, or which are in the middle of a *vfork*(2) are excepted and allowed to produce output.

Summary of modes.

Unfortunately, due to the evolution of the terminal driver, there are 4 different structures which contain various portions of the driver data. The first of these (**sgttyb**) contains that part of the information largely common between version 6 and version 7 UNIX systems. The second contains additional control characters added in version 7. The third is a word

of local state peculiar to the new terminal driver, and the fourth is another structure of special characters added for the new driver. In the future a single structure may be made available to programs which need to access all this information; most programs need not concern themselves with all this state.

Basic modes: sgtty.

The basic *ioctl*s use the structure defined in *<sgtty.h>*:

```
struct sgttyb {
        char    sg_ispeed;
        char    sg_ospeed;
        char    sg_erase;
        char    sg_kill;
        short   sg_flags;
};
```

The *sg_ispeed* and *sg_ospeed* fields describe the input and output speeds of the device according to the following table, which corresponds to the DEC DH-11 interface. If other hardware is used, impossible speed changes are ignored. Symbolic values in the table are as defined in *<sgtty.h>*.

B0	0	(hang up dataphone)
B50	1	50 baud
B75	2	75 baud
B110	3	110 baud
B134	4	134.5 baud
B150	5	150 baud
B200	6	200 baud
B300	7	300 baud
B600	8	600 baud
B1200	9	1200 baud
B1800	10	1800 baud
B2400	11	2400 baud
B4800	12	4800 baud
B9600	13	9600 baud
EXTA	14	External A
EXTB	15	External B

In the current configuration, only 110, 150, 300 and 1200 baud are really supported on dial-up lines. Code conversion and line control required for IBM 2741's (134.5 baud) must be implemented by the user's program. The half-duplex line discipline required for the 202 dataset (1200 baud) is not supplied; full-duplex 212 datasets work fine.

The *sg_erase* and *sg_kill* fields of the argument structure specify the erase and kill characters respectively. (Defaults are # and @.)

The *sg_flags* field of the argument structure contains several bits that determine the system's treatment of the terminal:

ALLDELAY	0177400	Delay algorithm selection
BSDELAY	0100000	Select backspace delays (not implemented):
BS0	0	
BS1	0100000	
VTDELAY	0040000	Select form-feed and vertical-tab delays:
FF0	0	
FF1	0100000	
CRDELAY	0030000	Select carriage-return delays:
CR0	0	
CR1	0010000	
CR2	0020000	
CR3	0030000	
TBDELAY	0006000	Select tab delays:
TAB0	0	
TAB1	0001000	
TAB2	0004000	
XTABS	0006000	
NLDELAY	0001400	Select new-line delays:
NL0	0	
NL1	0000400	
NL2	0001000	
NL3	0001400	
EVENP	0000200	Even parity allowed on input (most terminals)
ODDP	0000100	Odd parity allowed on input
RAW	0000040	Raw mode: wake up on all characters, 8-bit interface
CRMOD	0000020	Map CR into LF; echo LF or CR as CR-LF
ECHO	0000010	Echo (full duplex)
LCASE	0000004	Map upper case to lower on input
CBREAK	0000002	Return each character as soon as typed
TANDEM	0000001	Automatic flow control

The delay bits specify how long transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. In all cases a value of 0 indicates no delay.

Backspace delays are currently ignored but might be used for Terminet 300's.

If a form-feed/vertical tab delay is specified, it lasts for about 2 seconds.

Carriage-return delay type 1 lasts about .08 seconds and is suitable for the Terminet 300. Delay type 2 lasts about .16 seconds and is suitable for the VT05 and the TI 700. Delay type 3 is suitable for the concept-100 and pads lines to be at least 9 characters at 9600 baud.

New-line delay type 1 is dependent on the current column and is tuned for Teletype model 37's. Type 2 is useful for the VT05 and is about .10 seconds. Type 3 is unimplemented and is 0.

Tab delay type 1 is dependent on the amount of movement and is tuned to the Teletype model 37. Type 3, called XTABS, is not a delay at all but causes tabs to be replaced by the appropriate number of spaces on output.

Input characters with the wrong parity, as determined by bits 200 and 100, are ignored in cooked and CBREAK mode.

RAW disables all processing save output flushing with LFLUSHO; full 8 bits of input are given as soon as it is available; all 8 bits are passed on output. A break condition in the input is reported as a null character. If the input queue overflows in raw mode it is discarded; this applies to both new and old drivers.

CRMOD causes input carriage returns to be turned into new-lines; input of either CR or LF causes LF-CR both to be echoed (for terminals with a new-line function).

CBREAK is a sort of half-cooked (rare?) mode. Programs can read each character as soon as typed, instead of waiting for a full line; all processing is done except the input editing: character and word erase and line kill, input reprint, and the special treatment of \ or EOT are disabled.

TANDEM mode causes the system to produce a stop character (default ^S) whenever the input queue is in danger of overflowing, and a start character (default ^Q) when the input queue has drained sufficiently. It is useful for flow control when the 'terminal' is really another computer which understands the conventions.

Basic ioctls

In addition to the TIOCSETD and TIOCGETD disciplines discussed in **Line disciplines** above, a large number of other *ioctl*(2) calls apply to terminals, and have the general form:

#include <sgtty.h>

ioctl(fildes, code, arg)
struct sgttyb *arg;

The applicable codes are:

TIOCGETP Fetch the basic parameters associated with the terminal, and store in the pointed-to *sgttyb* structure.

TIOCSETP Set the parameters according to the pointed-to *sgttyb* structure. The interface delays until output is quiescent, then throws away any unread characters, before changing the modes.

TIOCSETN Set the parameters like TIOCSETP but do not delay or flush input. Input is not preserved, however, when changing to or from RAW.

With the following codes the *arg* is ignored.

TIOCEXCL Set "exclusive-use" mode: no further opens are permitted until the file has been closed.

TIOCNXCL Turn off "exclusive-use" mode.

TIOCHPCL When the file is closed for the last time, hang up the terminal. This is useful when the line is associated with an ACU used to place outgoing calls.

TIOCFLUSH All characters waiting in input or output queues are flushed.

The remaining calls are not available in vanilla version 7 UNIX. In cases where arguments are required, they are described; *arg* should otherwise be given as 0.

TIOCSTI the argument is the address of a character which the system pretends was typed on the terminal.

TIOCSBRK the break bit is set in the terminal.

TIOCCBRK the break bit is cleared.

TIOCSDTR data terminal ready is set.

TIOCCDTR data terminal ready is cleared.

TIOCGPGRP arg is the address of a word into which is placed the process group number of the control terminal.

TIOCSPGRP arg is a word (typically a process id) which becomes the process group for the control terminal.

FIONREAD returns in the long integer whose address is arg the number of immediately readable characters from the argument unit. This works for files, pipes, and terminals, but not (yet) for multiplexed channels.

Tchars

The second structure associated with each terminal specifies characters that are special in both the old and new terminal interfaces: The following structure is defined in *<sys/ioctl.h>*, which is automatically included in *<sgtty.h>*:

```
struct tchars {
        char    t_intrc;        /* interrupt */
        char    t_quitc;        /* quit */
        char    t_startc;       /* start output */
        char    t_stopc;        /* stop output */
        char    t_eofc;         /* end-of-file */
        char    t_brkc;         /* input delimiter (like nl) */
};
```

The default values for these characters are ^?, ^\, ^Q, ^S, ^D, and −1. A character value of −1 eliminates the effect of that character. The *t_brkc* character, by default −1, acts like a new-line in that it terminates a 'line,' is echoed, and is passed to the program. The 'stop' and 'start' characters may be the same, to produce a toggle effect. It is probably

counterproductive to make other special characters (including erase and kill) identical. The applicable ioctl calls are:

TIOCGETC
Get the special characters and put them in the specified structure.

TIOCSETC
Set the special characters to those given in the structure.

Local mode

The third structure associated with each terminal is a local mode word; except for the LNOHANG bit, this word is interpreted only when the new driver is in use. The bits of the local mode word are:

LCRTBS	000001	Backspace on erase rather than echoing erase
LPRTERA	000002	Printing terminal erase mode
LCRTERA	000004	Erase character echoes as backspace-space-backspace
LTILDE	000010	Convert ~ to ` on output (for Hazeltine terminals)
LMDMBUF	000020	Stop/start output when carrier drops
LLITOUT	000040	Suppress output translations
LTOSTOP	000100	Send SIGTTOU for background output
LFLUSHO	000200	Output is being flushed
LNOHANG	000400	Don't send hangup when carrier drops
LETXACK	001000	Diablo style buffer hacking (unimplemented)
LCRTKIL	002000	BS-space-BS erase entire line on line kill
LINTRUP	004000	Generate interrupt SIGTINT when input ready to read
LCTLECH	010000	Echo input control chars as ^X, delete as ^?
LPENDIN	020000	Retype pending input at next read or input character
LDECCTQ	040000	Only ^Q restarts output after ^S, like DEC systems

The applicable *ioctl* functions are:

TIOCLBIS arg is the address of a mask which is the bits to be set in the local mode word.

TIOCLBIC arg is the address of a mask of bits to be cleared in the local mode word.

TIOCLSET arg is the address of a mask to be placed in the local mode word.

TIOCLGET arg is the address of a word into which the current mask is placed.

Local special chars

The final structure associated with each terminal is the *ltchars* structure which defines interrupt characters for the new terminal driver. Its structure is:

```
struct ltchars {
        char    t_suspc;        /* stop process signal */
        char    t_dsuspc;       /* delayed stop process signal */
```

```
        char    t_rprntc;       /* reprint line */
        char    t_flushc;       /* flush output (toggles) */
        char    t_werasc;       /* word erase */
        char    t_lnextc;       /* literal next character */
};
```

The default values for these characters are ^Z, ^Y, ^R, ^O, ^W, and ^V. A value of −1 disables the character.

The applicable *ioctl* functions are:

TIOCSLTC
args is the address of a *ltchars* structure which defines the new local special characters.

TIOCGLTC
args is the address of a *ltchars* structure into which is placed the current set of local special characters.

FILES

/dev/tty
/dev/tty*
/dev/console

SEE ALSO

csh(1), stty(1), ioctl(2), sigvec(2), stty(3C), getty(8), init(8)

RESTRICTIONS

Half-duplex terminals are not supported.

STATUS

TTY(4) is supported by Digital Equipment Corporation.

NAME

tu – VAX-11/730 and VAX-11/750 TU58 console cassette interface

SYNTAX

options MRSP (for VAX-11/750's with an MRSP prom)

DESCRIPTION

The *tu* interface provides access to the VAX 11/730 and 11/750 TU58 console cassette drive(s).

The interface supports only block i/o to the TU58 cassettes. The devices are normally manipulated with the *arff*(8V) program using the "f" and "m" options.

The device driver is automatically included when a system is configured to run on an 11/730 or 11/750.

The TU58 on an 11/750 uses the Radial Serial Protocol (RSP) to communicate with the cpu over a serial line. This protocol is inherently unreliable as it has no flow control measures built in. On an 11/730 the Modified Radial Serial Protocol is used. This protocol incorporates flow control measures which insure reliable data transfer between the cpu and the device. Certain 11/750's have been modified to use the MRSP prom used in the 11/730. To reliably use the console TU58 on an 11/750 under UNIX, the MRSP prom is required. For those 11/750's without an MRSP prom, an unreliable but often useable interface has been developed. This interface uses an assembly language "pseudo-dma" routine to minimize the receiver interrupt service latency. To include this code in the system, the configuration must **not** specify the system will run on an 11/730 or use an MRSP prom. This unfortunately makes it impossible to configure a single system which will properly handle TU58's on both an 11/750 and an 11/730 (unless both machines have MRSP proms).

FILES

/dev/tu0
/dev/tu1 (only on a VAX-11/730)

SEE ALSO

arff(8V)

DIAGNOSTICS

tu%d: no bp, active %d. A transmission complete interrupt was received with no outstanding i/o request. This indicates a hardware problem.

tu%d protocol error, state=%s, op=%x, cnt=%d, block=%d. The driver entered an illegal state. The information printed indicates the illegal state, operation currently being executed, the i/o count, and the block number on the cassette.

tu%d receive state error, state=%s, byte=%x. The driver entered an illegal state in the receiver finite state machine. The state is shown along with the control byte of the received packet.

tu%d: read stalled. A timer watching the controller detected no interrupt for an extended period while an operation was outstanding. This usually indicates that one or more receiver interrupts were lost and the transfer is restarted (11/750 only).

tu%d: hard error bn%d, pk_mod %o. The device returned a status code indicating a hard error. The actual error code is shown in octal. No retries are attempted by the driver.

RESTRICTIONS

The VAX-11/750 console interface without MRSP prom is unuseable while the system is multi-user; it should be used only with the system running single-user and, even then, with caution.

STATUS

TU(4) is supported by Digital Equipment Corporation.

NAME

uda – UDA-50 disk controller interface

SYNTAX

controller uda0 at uba0 csr 0172150 vector udintr
disk ra0 at uda0 drive 0

DESCRIPTION

This is a driver for the DEC UDA-50 disk controller. The UDA-50 communicates with the host through a packet oriented protocol termed the Mass Storage Control Protocol (MSCP). Consult the file *<vax/mscp.h>* for a detailed description of this protocol.

Files with minor device numbers 0 through 7 refer to various portions of drive 0; minor devices 8 through 15 refer to drive 1, etc. The standard device names begin with "ra" followed by the drive number and then a letter a-h for partitions 0-7 respectively. The character ? stands here for a drive number in the range 0-7.

The block files access the disk via the system's normal buffering mechanism and may be read and written without regard to physical disk records. There is also a 'raw' interface which provides for direct transmission between the disk and the user's read or write buffer. A single read or write call results in exactly one I/O operation and therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw files conventionally begin with an extra 'r.'

In raw I/O counts should be a multiple of 512 bytes (a disk sector). Likewise *seek* calls should specify a multiple of 512 bytes.

DISK SUPPORT

This driver handles all drives which may be connected to the UDA-50. Drive types per se are not recognized, but rather the variable length partitions are defined as having an "infinite" length and the controller is relied on to return an error when an inaccessible block is requested. For constructing file systems, however the partitions sizes are required. The origin and size (in sectors) of the pseudo-disks on each drive are shown below. Partitions are not rounded to cylinder boundaries, as on other drives, because the type of drive attached to the controller is discovered too late in the autoconfiguration process to maintain separate partition tables for each drive. (The lack of proper drive type recognition would be more easily dealt with if the partition tables were read off the drive.)

RA60 partitions

disk	start	length
ra?a	0	15884
ra?b	15884	33440
ra?c	0	400176
ra?g	49324	82080
ra?h	131404	268772

RA80 partitions

disk	start	length
ra?a	0	15884

ra?b	15884	33440
ra?c	0	242606
ra?g	49324	82080
ra?h	131404	111202

RA81 partitions

disk	start	length
ra?a	0	15884
ra?b	15884	33440
ra?c	0	891072
ra?d	340670	15884
ra?e	356554	55936
ra?f	412490	478582
ra?g	49324	82080
ra?h	131404	759668

The ra?a partition is normally used for the root file system, the ra?b partition as a paging area, and the ra?c partition for pack-pack copying (it maps the entire disk).

FILES

/dev/ra[0-9][a-f]
/dev/rra[0-9][a-f]

DIAGNOSTICS

The following messages are printed at the console:

uda%d: random interrupt ignored
An unexpected interrupt was received (e.g. when no i/o was pending). The interrupt is ignored.

uda%d: interrupt in unknown state %d ignored
An interrupt was received when the driver was in an unknown internal state. Indicates a hardware problem or a driver bug.

uda%d: fatal error (%o)
The UDA-50 indicated a "fatal error" in the status returned to the host. The contents of the status register are displayed.

OFFLINE
(Additional status information given after a hard i/o error.) A hard i/o error occurred because the drive was not on-line.

uda: unknown packet
An MSCP packet of unknown type was received from the UDA-50. Check the cabling to the controller.

Udaerror udasa (%x)

error: com %d opc 0x%x stat 0x%x

uda%d: hard error

udintr: um ubinfo == 0

endcd=%o, stat=%o

Uda(%d) udasa (%x)

Uda (%d) Error (%x)

The following messages are written to the system error log file, /usr/adm/messages.

udastrat: ubinfo 0x%x
(VAX 11/750 only.) When allocating UNIBUS resources, the driver found it already had resources previously allocated.

Uda%d udasa %o, state %d
(Additional status information given after a hard i/o error.) The values of the UDA-50 status register and the internal driver state are printed.

The following errors are interpretations of MSCP error messages returned by the UDA-50 to the host.

controller error, event 0%o

host memory access error, event 0%o, addr 0%o

disk transfer error, unit %d

SDI error, unit %d, event 0%o

small disk error, unit %d, event 0%o, cyl %d

unknown error, unit %d, format 0%o, event 0%o

uda%d: %s error

STATUS

UDA(4) is supported by Digital Equipment Corporation.

NAME

udp – Internet User Datagram Protocol

SYNTAX

#include <sys/socket.h>
#include <netinet/in.h>

s = socket(AF_INET, SOCK_DGRAM, 0);

DESCRIPTION

UDP is a simple, unreliable datagram protocol which is used to support the SOCK DGRAM abstraction for the Internet protocol family. UDP sockets are connectionless, and are normally used with the *sendto* and *recvfrom* calls, though the *connect*(2) call may also be used to fix the destination for future packets (in which case the *recv*(2) or *read*(2) and *send*(2) or *write(2)* system calls may be used).

UDP address formats are identical to those used by TCP. In particular UDP provides a port identifier in addition to the normal Internet address format. Note that the UDP port space is separate from the TCP port space (i.e. a UDP port may not be "connected" to a TCP port). In addition broadcast packets may be sent (assuming the underlying network supports this) by using a reserved "broadcast address"; this address is network interface dependent.

DIAGNOSTICS

A socket operation may fail with one of the following errors returned:

[EISCONN] when trying to establish a connection on a socket which already has one, or when trying to send a datagram with the destination address specified and the socket is already connected;

[ENOTCONN] when trying to send a datagram, but no destination address is specified, and the socket hasn't been connected;

[ENOBUFS] when the system runs out of memory for an internal data structure;

[EADDRINUSE] when an attempt is made to create a socket with a port which has already been allocated;

[EADDRNOTAVAIL] when an attempt is made to create a socket with a network address for which no network interface exists.

SEE ALSO

send(2), recv(2), intro(4N), inet(4F)

STATUS

UDP(4P) currently is not supported by Digital Equipment Corporation.

NAME

un – Ungermann-Bass interface

SYNTAX

device un0 at uba0 csr 0160210 vector unintr

DESCRIPTION

The *un* interface provides access to a 4 Mb/s baseband network. The hardware uses a standard DEC DR11-W DMA interface in communicating with the host. The Ungermann-Bass hardware incorporates substantial protocol software in the network device in an attempt to offload protocol processing from the host.

The network number on which the interface resides must be specified at boot time with an SIOCSIFADDR ioctl. The host's address is discovered by communicating with the interface. The interface will not transmit or receive any packets before the network number has been defined.

DIAGNOSTICS

un%d: can't initialize. Insufficient UNIBUS resources existed for the device to complete initialization. Usually caused by having multiple network interfaces configured using buffered data paths on a data path poor machine such as the 11/750.

un%d: unexpected reset. The controller indicated a reset when none had been requested. Check the hardware (but see the bugs section below).

un%d: stray interrupt. An unexpected interrupt was received. The interrupt was ignored.

un%d: input error csr=%b. The controller indicated an error on moving data from the device to host memory.

un%d: bad packet type %d. A packet was received with an unknown packet type. The packet is discarded.

un%d: output error csr=%b. The device indicated an error on moving data from the host to device memory.

un%d: invalid state %d csr=%b. The driver found itself in an invalid internal state. The state is reset to a base state.

un%d: can't handle af%d. A request was made to send a message with an address format which the driver does not understand. The message is discarded and an error is returned to the user.

un%d: error limit exceeded. Too many errors were encountered in normal operation. The driver will attempt to reset the device, desist from attempting any i/o for approximately 60 seconds, then reset itself to a base state in hopes of resyncing itself up with the hardware.

un%d: restarting. After exceeding its error limit and resetting the device, the driver is restarting operation.

SEE ALSO

intro(4N), inet(4F)

RESTRICTIONS

The device does not reset itself properly resulting in the interface getting hung up in a state from which the only recourse is to reboot the system.

STATUS

UN(4) currently is not supported by Digital Equipment Corporation.

NAME

up – unibus storage module controller/drives

SYNTAX

controller sc0 at uba? csr 0176700 vector upintr
disk up0 at sc0 drive 0

DESCRIPTION

This is a generic UNIBUS storage module disk driver. It is specifically designed to work with the Emulex SC-21 controller. It can be easily adapted to other controllers (although bootstrapping will not necessarily be directly possible.)

Files with minor device numbers 0 through 7 refer to various portions of drive 0; minor devices 8 through 15 refer to drive 1, etc. The standard device names begin with "up" followed by the drive number and then a letter a-h for partitions 0-7 respectively. The character ? stands here for a drive number in the range 0-7.

The block files access the disk via the system's normal buffering mechanism and may be read and written without regard to physical disk records. There is also a 'raw' interface which provides for direct transmission between the disk and the user's read or write buffer. A single read or write call results in exactly one I/O operation and therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw files conventionally begin with an extra 'r.'

In raw I/O counts should be a multiple of 512 bytes (a disk sector). Likewise *seek* calls should specify a multiple of 512 bytes.

DISK SUPPORT

The driver interrogates the controller's holding register to determine the type of drive attached. The driver recognizes four different drives: AMPEX 9300, CDC 9766, AMPEX Capricorn, and FUJITSU 160. The origin and size of the pseudo-disks on each drive are as follows:

CDC 9766 300M drive partitions:

disk	start	length	cyl
up?a	0	15884	0-26
up?b	16416	33440	27-81
up?c	0	500384	0-822
up?d	341696	15884	562-588
up?e	358112	55936	589-680
up?f	414048	861760	681-822
up?g	341696	158528	562-822
up?h	49856	291346	82-561

AMPEX 9300 300M drive partitions:

disk	start	length	cyl
up?a	0	15884	0-26
up?b	16416	33440	27-81
up?c	0	495520	0-814

up?d	341696	15884	562-588
up?e	358112	55936	589-680
up?f	414048	81312	681-814
up?g	341696	153664	562-814
up?h	49856	291346	82-561

AMPEX Capricorn 330M drive partitions:

disk	start	length	cyl
hp?a	0	15884	0-31
hp?b	16384	33440	32-97
hp?c	0	524288	0-1023
hp?d	342016	15884	668-699
hp?e	358400	55936	700-809
hp?f	414720	109408	810-1023
hp?g	342016	182112	668-1023
hp?h	50176	291346	98-667

FUJITSU 160M drive partitions:

disk	start	length	cyl
up?a	0	15884	0-49
up?b	16000	33440	50-154
up?c	0	263360	0-822
up?d	49600	15884	155-204
up?e	65600	55936	205-379
up?f	121600	141600	380-822
up?g	49600	213600	155-822

It is unwise for all of these files to be present in one installation, since there is overlap in addresses and protection becomes a sticky matter. The up?a partition is normally used for the root file system, the up?b partition as a paging area, and the up?c partition for pack-pack copying (it maps the entire disk). On 160M drives the up?g partition maps the rest of the pack. On other drives both up?g and up?h are used to map the remaining cylinders.

FILES

/dev/up[0-7][a-h] block files
/dev/rup[0-7][a-h] raw files

SEE ALSO

hk(4), hp(4), uda(4)

DIAGNOSTICS

up%d%c: hard error sn%d cs2=%b er1=%b er2=%b. An unrecoverable error occurred during transfer of the specified sector in the specified disk partition. The contents of the cs2, er1 and er2 registers are printed in octal and symbolically with bits decoded. The error was either unrecoverable, or a large number of retry attempts (including offset positioning and drive recalibration) could not recover the error.

up%d: write locked. The write protect switch was set on the drive when a write was attempted. The write operation is not recoverable.

up%d: not ready. The drive was spun down or off line when it was accessed. The i/o operation is not recoverable.

up%d: not ready (flakey). The drive was not ready, but after printing the message about being not ready (which takes a fraction of a second) was ready. The operation is recovered if no further errors occur.

up%d%c: soft ecc sn%d. A recoverable ECC error occurred on the specified sector of the specified disk partition. This happens normally a few times a week. If it happens more frequently than this the sectors where the errors are occurring should be checked to see if certain cylinders on the pack, spots on the carriage of the drive or heads are indicated.

sc%d: lost interrupt. A timer watching the controller detecting no interrupt for an extended period while an operation was outstanding. This indicates a hardware or software failure. There is currently a hardware/software problem with spinning down drives while they are being accessed which causes this error to occur. The error causes a UNIBUS reset, and retry of the pending operations. If the controller continues to lose interrupts, this error will recur a few seconds later.

RESTRICTIONS

In raw I/O *read* and *write*(2) truncate file offsets to 512-byte block boundaries, and *write* scribbles on the tail of incomplete blocks. Thus, in programs that are likely to access raw devices, *read, write* and *lseek*(2) should always deal in 512-byte multiples.

STATUS

UP(4) currently is not supported by Digital Equipment Corporation.

NAME

ut – UNIBUS TU45 tri-density tape drive interface

SYNTAX

controller ut0 at uba0 csr 0172440 vector utintr
tape tj0 at ut0 drive 0

DESCRIPTION

The *ut* interface provides access to a standard tape drive interface as describe in *mtio*(4). Hardware implementing this on the VAX is typified by the System Industries SI 9700 tape subsystem. Tapes may be read or written at 800, 1600, and 6250 bpi.

SEE ALSO

mt(1), mtio(4)

DIAGNOSTICS

tj%d: no write ring. An attempt was made to write on the tape drive when no write ring was present; this message is written on the terminal of the user who tried to access the tape.

tj%d: not online. An attempt was made to access the tape while it was offline; this message is written on the terminal of the user who tried to access the tape.

tj%d: can't change density in mid-tape. An attempt was made to write on a tape at a different density than is already recorded on the tape. This message is written on the terminal of the user who tried to switch the density.

ut%d: soft error bn%d cs1=%b er=%b cs2=%b ds=%b. The formatter indicated a corrected error at a density other than 800bpi. The data transferred is assumed to be correct.

ut%d: hard error bn%d cs1=%b er=%b cs2=%b ds=%b. A tape error occurred at block *bn*. Any error is fatal on non-raw tape; when possible the driver will have retried the operation which failed several times before reporting the error.

tj%d: lost interrupt. A tape operation did not complete within a reasonable time, most likely because the tape was taken off-line during rewind or lost vacuum. The controller should, but does not, give an interrupt in these cases. The device will be made available again after this message, but any current open reference to the device will return an error as the operation in progress aborts.

RESTRICTIONS

If any non-data error is encountered on non-raw tape, it refuses to do anything more until closed.

STATUS

UT(4) currently is not supported by Digital Equipment Corporation.

NAME

uu – TU58/DECtape II UNIBUS cassette interface

SYNTAX

options UUDMA
device uu0 at uba0 csr 0176500 vector uurintr uuxintr

DESCRIPTION

The *uu* device provides access to dual DEC TU58 tape cartridge drives connected to the UNIBUS via a DL11-W interface module.

The interface supports only block i/o to the TU58 cassettes. The drives are normally manipulated with the *arff*(8V) program using the "m" and "f" options.

The driver provides for an optional write and verify (read after write) mode that is activated by specifying the "a" device.

The TU58 is treated as a single device by the system even though it has two separate drives, "uu0" and "uu1". If there is more than one TU58 unit on a system, the extra drives are named "uu2", "uu3" etc.

NOTES

Assembly language code to assist the driver in handling the receipt of data (using a pseudo-dma approach) should be included when using this driver; specify "options UUDMA" in the configuration file.

ERRORS

The following errors may be returned:

[ENXIO] Nonexistent drive (on open); offset is too large or bad (undefined) ioctl code.

[EIO] Open failed, the device could not be reset.

[EBUSY] Drive in use.

FILES

/dev/uu?
/dev/uu?a

SEE ALSO

tu(4), arff(8V)

DIAGNOSTICS

uu%d: no bp, active %d. A transmission complete interrupt was received with no outstanding i/o request. This indicates a hardware problem.

uu%d protocol error, state=%s, op=%x, cnt=%d, block=%d. The driver entered an illegal state. The information printed indicates the illegal state, the operation currently being executed, the i/o count, and the block number on the cassette.

uu%d: break received, transfer restarted. The TU58 was sending a continuous break signal and had to be reset. This may indicate a hardware problem, but the driver will attempt to recover from the error.

uu%d receive state error, state=%s, byte=%x. The driver entered an illegal state in the receiver finite state machine. The state is shown along with the control byte of the received packet.

uu%d: read stalled. A timer watching the controller detected no interrupt for an extended period while an operation was outstanding. This usually indicates that one or more receiver interrupts were lost and the transfer is restarted.

uu%d: hard error bn%d, pk_mod %o. The device returned a status code indicating a hard error. The actual error code is shown in octal. No retries are attempted by the driver.

STATUS

UU(4) currently is notsupported by Digital Equipment Corporation.

NAME

va – Benson-Varian interface

SYNTAX

controller va0 at uba0 csr 0164000 vector vaintr
disk vz0 at va0 drive 0

DESCRIPTION

(NOTE: the configuration description, while counter-intuitive, is actually as shown above.)

The Benson-Varian printer/plotter in normally used with the programs *vpr*(1), *vprint*(1) or *vtroff*(1). This description is designed for those who wish to drive the Benson-Varian directly.

In print mode, the Benson-Varian uses a modified ASCII character set. Most control characters print various non-ASCII graphics such as daggers, sigmas, copyright symbols, etc. Only LF and FF are used as format effectors. LF acts as a newline, advancing to the beginning of the next line, and FF advances to the top of the next page.

In plot mode, the Benson-Varian prints one raster line at a time. An entire raster line of bits (2112 bits = 264 bytes) is sent, and then the Benson-Varian advances to the next raster line.

Note: The Benson-Varian must be sent an even number of bytes. If an odd number is sent, the last byte will be lost. Nulls can be used in print mode to pad to an even number of bytes.

To use the Benson-Varian yourself, you must realize that you cannot open the device, */dev/va0* if there is a daemon active. You can see if there is an active daemon by doing a *lpq*(1) and seeing if there are any files being printed.

To set the Benson-Varian into plot mode include the file *<sys/vcmd.h>* and use the following *ioctl*(2) call

```
ioctl(fileno(va), VSETSTATE, plotmd);
```

where *plotmd* is defined to be

```
int plotmd[] = { VPLOT, 0, 0 };
```

and *va* is the result of a call to *fopen* on stdio. When you finish using the Benson-Varian in plot mode you should advance to a new page by sending it a FF after putting it back into print mode, i.e. by

```
int prtmd[] = { VPRINT, 0, 0 };
...
fflush(va);
ioctl(fileno(va), VSETSTATE, prtmd);
write(fileno(va), "\f\0", 2);
```

N.B.: If you use the standard I/O library with the Benson-Varian you **must** do

```
setbuf(vp, vpbuf);
```

where *vpbuf* is declared

```
char vpbuf[BUFSIZ];
```

otherwise the standard I/O library, thinking that the Benson-Varian is a terminal (since it is a character special file) will not adequately buffer the data you are sending to the Benson-Varian. This will cause it to run **extremely** slowly and tend to grind the system to a halt.

FILES

/dev/va0

SEE ALSO

vfont(5), lpr(1), lpd(8), vtroff(1), vp(4)

DIAGNOSTICS

The following error numbers are significant at the time the device is opened.

[ENXIO] The device is already in use.

[EIO] The device is offline.

The following message may be printed on the console.

va%d: npr timeout. The device was not able to get data from the UNIBUS within the timeout period, most likely because some other device was hogging the bus.

STATUS

VA(4) currently is not supported by Digital Equipment Corporation.

NAME

vp – Versatec interface

SYNTAX

device vp0 at uba0 csr 0177510 vector vpintr vpintr

DESCRIPTION

The Versatec printer/plotter is normally used with the programs *vpr*(1), *vprint*(1) or *vtroff*(1). This description is designed for those who wish to drive the Versatec directly.

To use the Versatec yourself, you must realize that you cannot open the device, */dev/vp0* if there is a daemon active. You can see if there is a daemon active by doing a *lpq*(1), and seeing if there are any files being sent.

To set the Versatec into plot mode you should include *<sys/vcmd.h>* and use the *ioctl*(2) call

```
ioctl(fileno(vp), VSETSTATE, plotmd);
```

where *plotmd* is defined to be

```
int plotmd[] = { VPLOT, 0, 0 };
```

and *vp* is the result of a call to *fopen* on stdio. When you finish using the Versatec in plot mode you should eject paper by sending it a EOT after putting it back into print mode, i.e. by

```
int prtmd[] = { VPRINT, 0, 0 };
...
fflush(vp);
ioctl(fileno(vp), VSETSTATE, prtmd);
write(fileno(vp), "\04", 1);
```

N.B.: If you use the standard I/O library with the Versatec you **must** do

```
setbuf(vp, vpbuf);
```

where *vpbuf* is declared

```
char vpbuf[BUFSIZ];
```

otherwise the standard I/O library, thinking that the Versatec is a terminal (since it is a character special file) will not adequately buffer the data you are sending to the Versatec. This will cause it to run **extremely** slowly and tends to grind the system to a halt.

FILES

/dev/vp0

SEE ALSO

vfont(5), lpr(1), lpd(8), vtroff(1), va(4)

DIAGNOSTICS

The following error numbers are significant at the time the device is opened.

[ENXIO] The device is already in use.

[EIO] The device is offline.

RESTRICTIONS

The configuration part of the driver assumes that the device is setup to vector print mode through 0174 and plot mode through 0200. Since the driver doesn't care whether the device considers the interrupt to be a print or a plot interrupt, it would be preferable to have these be the same. This since the configuration program can't be sure at boot time which vector interrupted and where the interrupt vectors actually are. For the time being, since our versatec is vectored as described above, we specify that it has two interrupt vectors and are careful to detect an interrupt through 0200 at boot time and (manually) pretend the interrupt came through 0174.

STATUS

VP(4) currently is not supported by Digital Equipment Corporation.

NAME

vv – Proteon proNET 10 Megabit ring

SYNTAX

device vv0 at uba0 csr 161000 vector vvrint vvxint

DESCRIPTION

The *vv* interface provides access to a 10 Mb/s Proteon proNET ring network.

The network number to which the interface is attached must be specified with an SIOCSIFADDR ioctl before data can be transmitted or received. The host's address is discovered by putting the interface in digital loopback mode (not joining the ring) and sending a broadcast packet from which the source address is extracted. the Internet address of the interface would be 128.3.0.24.

The interface software implements error-rate limiting on the input side. This provides a defense against situations where other hosts or interface hardware failures cause a machine to be inundated with garbage packets. The scheme involves an exponential backoff where the input side of the interface is disabled for longer and longer periods. In the limiting case, the interface is turned on every two minutes or so to see if operation can resume.

If the installation is running CTL boards which use the old broadcast address of 0 instead of the new address of 0xff, the define OLD_BROADCAST should be specified in the driver.

If the installation has a Wirecenter, the define WIRECENTER should be specified in the driver. **N.B.**: Incorrect definition of WIRECENTER can cause hardware damage.

The interface normally tries to use a "trailer" encapsulation to minimize copying data on input and output. This may be disabled, on a per-interface basis, by setting the IFF_NOTRAILERS flag with an SIOCSIFFLAGS ioctl.

DIAGNOSTICS

vv%d: host %d. The software announces the host address discovered during autoconfiguration.

vv%d: can't initialize. The software was unable to discover the address of this interface, so it deemed "dead" will not be enabled.

vv%d: error vvocsr=%b. The hardware indicated an error on the previous transmission.

vv%d: output timeout. The token timer has fired and the token will be recreated.

vv%d: error vvicsr=%b. The hardware indicated an error in reading a packet off the ring.

en%d: can't handle af%d. The interface was handed a message with addresses formatted in an unsuitable address family; the packet was dropped.

vv%d: vs_olen=%d. The ring output routine has been handed a message with a preposterous length. This results in an immediate *panic: vs_olen*.

SEE ALSO

intro(4N), inet(4F)

STATUS

VV(4) currently is not supported by Digital Equipment Corporation.

NAME

intro – introduction to system maintenance and operation commands

DESCRIPTION

This section contains information related to system operation and maintenance. In particular, commands used to create new file systems, *newfs*, *mkfs*, and verify the integrity of the file systems, *fsck*, *icheck*, *dcheck*, and *ncheck* are described here. The section *format* should be consulted when formatting disk packs. The section *crash* should be consulted in understanding how to interpret system crash dumps.

LIST OF PROGRAMS

Program	*Appears on Page*	*Description*
ac	ac.8	login accounting
accton	sa.8	system accounting
analyze	analyze.8	Virtual UNIX postmortem crash analyzer
arcv	arcv.8	convert archives to new format
arff	arff.8v	archiver and copier for floppy
bad144	bad144.8	read/write dec standard 144 bad sector information
badsect	badsect.8	create files to contain bad sectors
bugfiler	bugfiler.8	file bug reports in folders automatically
catman	catman.8	create the cat files for the manual
chown	chown.8	change owner
clri	clri.8	clear i-node
comsat	comsat.8c	biff server
config	config.8	build system configuration files
cron	cron.8	clock daemon
dcheck	dcheck.8	file system directory consistency check
diskpart	diskpart.8	calculate default disk partition sizes
dmesg	dmesg.8	collect system diagnostic messages to form error log
drtest	drtest.8	standalone disk test program
dump	dump.8	incremental file system dump
dumpfs	dumpfs.8	dump file system information
edquota	edquota.8	edit user quotas
fastboot	fastboot.8	reboot/halt the system without checking the disks
fasthalt	fastboot.8	reboot/halt the system without checking the disks
flcopy	arff.8v	archiver and copier for floppy
format	format.8v	how to format disk packs
fsck	fsck.8	file system consistency check and interactive repair
ftpd	ftpd.8c	DARPA Internet File Transfer Protocol server
gettable	gettable.8c	get NIC format host tables from a host
getty	getty.8	set terminal mode
halt	halt.8	stop the processor
htable	htable.8	convert NIC standard format host tables
icheck	icheck.8	file system storage consistency check
implog	implog.8c	IMP log interpreter

implogd	implogd.8c	IMP logger process
init	init.8	process control initialization
kgmon	kgmon.8	generate a dump of the operating systems profile buffers
lpc	lpc.8	line printer control program
lpd	lpd.8	line printer daemon
makedev	makedev.8	make system special files
makekey	makekey.8	generate encryption key
mkfs	mkfs.8	construct a file system
mklost+found	mklost+found.8	make a lost+found directory for fsck
mknod	mknod.8	build special file
mkproto	mkproto.8	construct a prototype file system
mount	mount.8	mount and dismount file system
ncheck	ncheck.8	generate names from i-numbers
newfs	newfs.8	construct a new file system
pac	pac.8	printer/ploter accounting information
pstat	pstat.8	print system facts
quot	quot.8	summarize file system ownership
quotacheck	quotacheck.8	file system quota consistency checker
quotaoff	quotaon.8	turn file system quotas on and off
quotaon	quotaon.8	turn file system quotas on and off
rc	rc.8	command script for auto-reboot and daemons
rdump	rdump.8c	file system dump across the network
reboot	reboot.8	UNIX bootstrapping procedures
renice	renice.8	alter priority of running processes
repquota	repquota.8	summarize quotas for a file system
restore	restore.8	incremental file system restore
rexecd	rexecd.8c	remote execution server
rlogind	rlogind.8c	remote login server
rmt	rmt.8c	remote magtape protocol module
route	route.8c	manually manipulate the routing tables
routed	routed.8c	network routing daemon
rrestore	rrestore.8c	restore a file system dump across the network
rshd	rshd.8c	remote shell server
rwhod	rwhod.8c	system status server
rxformat	rxformat.8v	format floppy disks
sa	sa.8	system accounting
savecore	savecore.8	save a core dump of the operating system
sendmail	sendmail.8	send mail over the internet
setifaddr	setifaddr.8c	set network interface address
shutdown	shutdown.8	close down the system at a given time
sticky	sticky.8	executable files with persistent text
swapon	swapon.8	specify additional device for paging and swapping
sync	sync.8	update the super block
syslog	syslog.8	log systems messages

telnetd	telnetd.8c	DARPA TELNET protocol server
tftpd	tftpd.8c	DARPA Trivial File Transfer Protocol server
trpt	trpt.8c	transliterate protocol trace
tunefs	tunefs.8	tune up an existing file system
umount	mount.8	mount and dismount file system
update	update.8	periodically update the super block
uuclean	uuclean.8c	uucp spool directory clean-up
uusnap	uusnap.8c	show snapshot of the UUCP system
vipw	vipw.8	edit the password file

NAME

ac – login accounting

SYNTAX

/etc/ac [**–w** wtmp] [**–p**] [**–d**] [people] ...

DESCRIPTION

Ac produces a printout giving connect time for each user who has logged in during the life of the current *wtmp* file. A total is also produced. **–w** is used to specify an alternate *wtmp* file. **–p** prints individual totals; without this option, only totals are printed. **–d** causes a printout for each midnight to midnight period. Any *people* will limit the printout to only the specified login names. If no *wtmp* file is given, */usr/adm/wtmp* is used.

The accounting file */usr/adm/wtmp* is maintained by *init* and *login*. Neither of these programs creates the file, so if it does not exist no connect-time accounting is done. To start accounting, it should be created with length 0. On the other hand if the file is left undisturbed it will grow without bound, so periodically any information desired should be collected and the file truncated.

FILES

/usr/adm/wtmp

SEE ALSO

init(8), sa(8), login(1), utmp(5).

STATUS

AC(8) currently is not supported by Digital Equipment Corporation.

NAME

analyze – Virtual UNIX postmortem crash analyzer

SYNTAX

/etc/analyze [**–s** swapfile] [**–f**] [**–m**] [**–d**] [**–D**] [**–v**] corefile [system]

DESCRIPTION

Analyze is the post-mortem analyzer for the state of the paging system. In order to use *analyze* you must arrange to get a image of the memory (and possibly the paging area) of the system after it crashes (see *crash*(8V)).

The *analyze* program reads the relevant system data structures from the core image file and indexing information from **/vmunix** (or the specified file) to determine the state of the paging subsystem at the point of crash. It looks at each process in the system, and the resources each is using in an attempt to determine inconsistencies in the paging system state. Normally, the output consists of a sequence of lines showing each active process, its state (whether swapped in or not), its *p0br*, and the number and location of its page table pages. Any pages which are locked while raw i/o is in progress, or which are locked because they are *intransit* are also printed. (Intransit text pages often diagnose as duplicated; you will have to weed these out by hand.)

The program checks that any pages in core which are marked as not modified are, in fact, identical to the swap space copies. It also checks for non-overlap of the swap space, and that the core map entries correspond to the page tables. The state of the free list is also checked.

Options to *analyze*:

–D causes the diskmap for each process to be printed.

–d causes the (sorted) paging area usage to be printed.

–f which causes the free list to be dumped.

–m causes the entire coremap state to be dumped.

–v (long unused) which causes a hugely verbose output format to be used.

In general, the output from this program can be confused by processes which were forking, swapping, or exiting or happened to be in unusual states when the crash occurred. You should examine the flags fields of relevant processes in the output of a *pstat*(8) to weed out such processes.

It is possible to look at the core dump with *adb* if you do

```
adb –k /vmunix /vmcore
```

FILES

/vmunix default system namelist

SEE ALSO

adb(1), ps(1), crash(8V), pstat(8)

DIAGNOSTICS

Various diagnostics about overlaps in swap mappings, missing swap mappings, page table entries inconsistent with the core map, incore pages which are marked clean but differ from disk-image copies, pages which are locked or intransit, and inconsistencies in the free list.

STATUS

ANALYZE(8) currently is not supported by Digital Equipment Corporation.

NAME

arcv – convert archives to new format

SYNTAX

/etc/arcv file ...

DESCRIPTION

Arcv converts archive files (see *ar*(1), *ar*(5)) from 32v and Third Berkeley editions to a new portable format. The conversion is done in place, and the command refuses to alter a file not in old archive format.

Old archives are marked with a magic number of 0177545 at the start; new archives have a first line "!<arch>".

FILES

/tmp/v*, temporary copy

SEE ALSO

ar(1), ar(5)

STATUS

ARCV(8) currently is not supported by Digital Equipment Corporation.

NAME

arff, flcopy – archiver and copier for floppy

SYNTAX

/etc/arff [key] [name ...]
/etc/flcopy [**−h**] [**−t***n*]

DESCRIPTION

Arff saves and restores files on the console floppy disk. Its actions are controlled by the *key* argument. The *key* is a string of characters containing at most one function letter and possibly one or more function modifiers. Other arguments to the command are file names specifying which files are to be dumped or restored.

Files names have restrictions, because of radix50 considerations. They must be in the form 1-6 alphanumerics followed by "." followed by 0-3 alphanumerics. Case distinctions are lost. Only the trailing component of a pathname is used.

The function portion of the key is specified by one of the following letters:

r The named files are replaced where found on the floppy, or added taking up the minimal possible portion of the first empty spot on the floppy.

x The named files are extracted from the floppy.

d The named files are deleted from the floppy. Arff will combine contiguous deleted files into one empty entry in the rt-11 directory.

t The names of the specified files are listed each time they occur on the floppy. If no file argument is given, all of the names on the floppy are listed.

The following characters may be used in addition to the letter which selects the function desired.

v The **v** (verbose) option, when used with the **t** function gives more information about the floppy entries than just the name.

f causes *arff* to use the next argument as the name of the archive instead of /dev/floppy.

m causes *arff* not to use the mapping algorithm employed in interleaving sectors around a floppy disk. In conjunction with the **f** option it may be used for extracting files from rt11 formatted cartridge disks, for example. It may also be used to speed up reading from and writing to rx02 floppy disks, by using the 'c' device instead of the 'b' device.

c causes *arff* to create a new directory on the floppy, effectively deleting all previously existing files.

Flcopy copies the console floppy disk (opened as '/dev/floppy') to a file created in the current directory, named "floppy", then prints the message "Change Floppy, hit return when done". Then *flcopy* copies the local file back out to the floppy disk.

The **−h** option to *flcopy* causes it to open a file named "floppy" in the current directory and copy it to */dev/floppy;* the **−t** option causes only the first *n* tracks to participate in a copy.

Arff may also be used with the console TU58 cassettes on the 11/730. To do so, the **m** key must be specified. Normally, the **f** key is also used.

FILES

/dev/floppy or /dev/rrx??
floppy (in current directory)

SEE ALSO

fl(4), rx(4), rxformat(8V)

RESTRICTIONS

Floppy errors are handled ungracefully.

STATUS

ARFF(8V) is supported by Digital Equipment Corporation.

NAME

bad144 – read/write dec standard 144 bad sector information

SYNTAX

/etc/bad144 [**−f**] disktype disk [sno [bad ...]]

DESCRIPTION

Bad144 can be used to inspect the information stored on a disk that is used by the disk drivers to implement bad sector forwarding. The format of the information is specified by DEC standard 144, as follows.

The bad sector information is located in the first 5 even numbered sectors of the last track of the disk pack. There are five identical copies of the information, described by the *dkbad* structure.

Replacement sectors are allocated starting with the first sector before the bad sector information and working backwards towards the beginning of the disk. A maximum of 126 bad sectors are supported. The position of the bad sector in the bad sector table determines which replacement sector it corresponds to. The bad sectors must be listed in ascending order.

The bad sector information and replacement sectors are conventionally only accessible through the "c" file system partition of the disk. If that partition is used for a file system, the user is responsible for making sure that it does not overlap the bad sector information or any replacement sectors.

The bad sector structure is as follows:

```
struct dkbad {
        long        bt.csn;                 /* cartridge serial number */
        u.short     bt.mbz;                 /* unused; should be 0 */
        u.short     bt.flag;                /* -1 => alignment cartridge */
        struct bt bad {
                    u.short bt.cyl;         /* cylinder number of bad sector */
                    u.short bt.trksec;      /* track and sector number */
        } bt.bad[126];
};
```

Unused slots in the *bt bad* array are filled with all bits set, a putatively illegal value.

Bad144 is invoked by giving a device type (e.g. rk07, rm03, rm05, etc.), and a device name (e.g. hk0, hp1, etc.). It reads the first sector of the last track of the corresponding disk and prints out the bad sector information. It may also be invoked giving a serial number for the pack and a list of bad sectors, and will then write the supplied information onto the same location. Note, however, that *bad144* does not arrange for the specified sectors to be marked bad in this case. This option should only be used to restore known bad sector information which was destroyed. It is necessary to reboot before the change will take effect.

If the disk is an RP06, Fujitsu Eagle, or Ampex Capricorn on a Massbus, the **−f** option may be used to mark the bad sectors as "bad". This can only be done safely when there is no other disk activity, preferably while running single-user. Otherwise, new bad sectors can be added only by running a formatter. Note that the order in which the sectors are listed determines which sectors used for replacements; if new sectors are being inserted into the list on a drive that is in use, care should be taken that replacements for existing bad sectors have the correct contents.

SEE ALSO

badsect(8), format(8V)

RESTRICTIONS

On an 11/750, the standard bootstrap drivers used to boot the system do not understand bad sectors, handle ECC errors, or the special SSE (skip sector) errors of RM80 type disks. This means that none of these errors can occur when reading the file /vmunix to boot. Sectors 0-15 of the disk drive must also not have any of these errors.

The drivers which write a system core image on disk after a crash do not handle errors; thus the crash dump area must be free of errors and bad sectors.

STATUS

BAD144(8) is supported by Digital Equipment Corporation.

NAME

badsect – create files to contain bad sectors

SYNTAX

/etc/badsect bbdir sector ...

DESCRIPTION

Badsect makes a file to contain a bad sector. Normally, bad sectors are made inaccessible by the standard formatter, which provides a forwarding table for bad sectors to the driver; see *bad144*(8) for details. If a driver supports the bad blocking standard it is much preferable to use that method to isolate bad blocks, since the bad block forwarding makes the pack appear perfect, and such packs can then be copied with *dd*(1). The technique used by this program is also less general than bad block forwarding, as *badsect* can't make amends for bad blocks in the i-list of file systems or in swap areas.

On some disks, adding a sector which is suddenly bad to the bad sector table currently requires the running of the standard DEC formatter. Thus to deal with a newly bad block or on disks where the drivers do not support the bad-blocking standard *badsect* may be used to good effect.

Badsect is used on a quiet file system in the following way: First mount the file system, and change to its root directory. Make a directory BAD there. Run *badsect* giving as argument the BAD directory followed by all the bad sectors you wish to add. (The sector numbers must be relative to the beginning of the file system, but this is not hard as the system reports relative sector numbers in its console error messages.) Then change back to the root directory, unmount the file system and run *fsck*(8) on the file system. The bad sectors should show up in two files or in the bad sector files and the free list. Have *fsck* remove files containing the offending bad sectors, but **do not** have it remove the BAD/*nnnnn* files. This will leave the bad sectors in only the BAD files.

Badsect works by giving the specified sector numbers in a *mknod*(2) system call, creating an illegal file whose first block address is the block containing bad sector and whose name is the bad sector number. When it is discovered by *fsck* it will ask "HOLD BAD BLOCK"? A positive response will cause *fsck* to convert the inode to a regular file containing the bad block.

SEE ALSO

bad144(8), fsck(8), format(8V)

DIAGNOSTICS

Badsect refuses to attach a block that resides in a critical area or is out of range of the file system. A warning is issued if the block is already in use.

RESTRICTIONS

If more than one sector which comprise a file system fragment are bad, you should specify only one of them to *badsect*, as the blocks in the bad sector files actually cover all the sectors in a file system fragment.

STATUS

BADSECT(8) is supported by Digital Equipment Corporation.

NAME

bugfiler – file bug reports in folders automatically

SYNTAX

bugfiler [mail directory]

DESCRIPTION

Bugfiler is a program to automatically intercept bug reports, summarize them and store them in the appropriate sub directories of the mail directory specified on the command line or the (system dependent) default. It is designed to be compatible with the Rand MH mail system. *Bugfiler* is normally invoked by the mail delivery program through *aliases*(5) with a line such as the following in /usr/lib/aliases.

bugs:"|bugfiler /usr/bugs/mail"

It reads the message from the standard input or the named file, checks the format and returns mail acknowledging receipt or a message indicating the proper format. Valid reports are then summarized and filed in the appropriate folder. Users can then log onto the system and check the summary file for bugs that pertain to them. Bug reports are submitted in RFC822 format and must contain the following header lines:

Date: <date the report is received>
From: <valid return address>
Subject: <short summary of the problem>
Index: <source directory>/<source file> <version> [Fix]

In addition, the body of the message must contain a line which begins with "Description:" followed by zero or more lines describing the problem in detail and a line beginning with "Repeat-By:" followed by zero or more lines describing how to repeat the problem. If the keyword 'Fix' is specified in the 'Index' line, then there must also be a line beginning with "Fix:" followed by a diff of the old and new source files or a description of what was done to fix the problem.

The 'Index' line is the key to the filing mechanism. The source directory name must match one of the folder names in the mail directory. The message is then filed in this folder and a line appended to the summary file in the following format:

<folder name>/<message number> <Index info>
<Subject info>

FILES

/usr/new/lib/mh/deliver	mail delivery program
/usr/new/lib/mh/unixtomh	converts unix mail format to mh format
maildir/.ack	the message sent in acknowledgement
maildir/.format	the message sent when format errors are detected
maildir/summary	the summary file
maildir/Bf??????	temporary copy of the input message
maildir/Rp??????	temporary file for the reply message.

SEE ALSO

mh(1), newaliases(1), aliases(5)

RESTRICTIONS

Since mail can be forwarded in a number of different ways, *bugfiler* does not recognize forwarded mail and will reply/complain to the forwarder instead of the original sender unless there is a 'Reply-To' field in the header.

Duplicate messages should be discarded or recognized and put somewhere else.

STATUS

BUGFILER(8) currently is not supported by Digital Equipment Corporation.

NAME

catman – create the cat files for the manual

SYNTAX

/etc/catman [**–p**] [**–n**] [**–w**] [sections]

DESCRIPTION

Catman creates the preformatted versions of the on-line manual from the nroff input files. Each manual page is examined and those whose preformatted versions are missing or out of date are recreated. If any changes are made, *catman* will recreate the **/usr/lib/whatis** database.

If there is one parameter not starting with a '–', it is take to be a list of manual sections to look in. For example

catman 123

will cause the updating to only happen to manual sections 1, 2, and 3.

Options:

–n prevents creations of **/usr/lib/whatis**.

–p prints what would be done instead of doing it.

–w causes only the **/usr/lib/whatis** database to be created. No manual reformatting is done.

FILES

/usr/man/man?/*.*	raw (nroff input) manual sections
/usr/man/cat?/*.*	preformatted manual pages
/usr/lib/makewhatis	commands to make whatis database

SEE ALSO

man(1)

STATUS

CATMAN(8) currently is not supported by Digital Equipment Corporation.

NAME

chown – change owner

SYNTAX

/etc/chown [**–f**] owner file ...

DESCRIPTION

Chown changes the owner of the *files* to *owner*. The owner may be either a decimal UID or a login name found in the password file.

Only the super-user can change owner, in order to simplify accounting procedures. No errors are reported when the **–f** (force) option is given.

FILES

/etc/passwd

SEE ALSO

chgrp(1), chown(2), passwd(5), group(5)

STATUS

CHOWN(8) is supported by Digital Equipment Corporation.

NAME

clri – clear i-node

SYNTAX

/etc/clri filesystem i-number ...

DESCRIPTION

Clri is obsoleted for normal file system repair work by *fsck*(8).

Clri writes zeros on the i-nodes with the decimal *i-numbers* on the *filesystem.* After *clri,* any blocks in the affected file will show up as 'missing' in an *icheck*(8) of the *filesystem.*

Read and write permission is required on the specified file system device. The i-node becomes allocatable.

The primary purpose of this routine is to remove a file which for some reason appears in no directory. If it is used to zap an i-node which does appear in a directory, care should be taken to track down the entry and remove it. Otherwise, when the i-node is reallocated to some new file, the old entry will still point to that file. At that point removing the old entry will destroy the new file. The new entry will again point to an unallocated i-node, so the whole cycle is likely to be repeated again and again.

SEE ALSO

icheck(8)

RESTRICTIONS

If the file is open, *clri* is likely to be ineffective.

STATUS

CLRI(8) is supported by Digital Equipment Corporation.

NAME

comsat – biff server

SYNTAX

/etc/comsat

DESCRIPTION

Comsat is the server process which listens for reports of incoming mail and notifies users if they have requested this service. *Comsat* listens on a datagram port associated with the "biff" service specification (see *services*(5)) for one line messages of the form

user@mailbox-offset

If the *user* specified is logged in to the system and the associated terminal has the owner execute bit turned on (by a "biff y"), the *offset* is used as a seek offset into the appropriate mailbox file and the first 7 lines or 560 characters of the message are printed on the user's terminal. Lines which appear to be part of the message header other than the "From", "To", "Date", or "Subject" lines are not included in the displayed message.

FILES

/etc/utmp to find out who's logged on and on what terminals

SEE ALSO

biff(1)

RESTRICTIONS

The message header filtering is prone to error.

STATUS

COMSAT(8C) currently is not supported by Digital Equipment Corporation.

NAME

config – build system configuration files

SYNTAX

/etc/config [**–p**] *config_file*

DESCRIPTION

Config builds a set of system configuration files from a short file which describes the sort of system that is being configured. It also takes as input a file which tells *config* what files are needed to generate a system. This can be augmented by a configuration specific set of files that give alternate files for a specific machine. (see the FILES section below) If the **–p** option is supplied, *config* will configure a system for profiling; c.f. *kgmon*(8), *gprof*(1).

Config should be run from the **conf** subdirectory of the system source (usually /sys/conf). *Config* assumes that there is already a directory *../config_file* created and it places all its output files in there. The output of *config* consists of a number files: **ioconf.c** contains a description of what i/o devices are attached to the system,; **ubglue.s** contains a set of interrupt service routines for devices attached to the UNIBUS; **makefile** is a file used by *make*(1) in building the system; a set of header files which contain the number of various devices that will be compiled into the system; and a set of swap configuration files which contain definitions for the disk areas to be used for swapping, the root file system, argument processing, and system dumps.

After running *config*, it is necessary to run "make depend" in the directory where the new makefile was created. *Config* reminds you of this when it completes.

If you get any other error messages from *config*, you should fix the problems in your configuration file and try again. If you try to compile a system that had configuration errors, you will likely meet with failure.

FILES

/sys/conf/makefile.vax	generic makefile for the VAX
/sys/conf/files	list of common files system is built from
/sys/conf/files.vax	list of VAX specific files
/sys/conf/devices.vax	name to major device mapping file for the VAX
/sys/conf/files.ERNIE	list of files specific to ERNIE system

SEE ALSO

"Building 4.2BSD UNIX System with Config"
The SYNTAX portion of each device in section 4.

RESTRICTIONS

The line numbers reported in error messages are usually off by one.

STATUS

CONFIG(8) is supported by Digital Equipment Corporation.

NAME

cron – clock daemon

SYNTAX

/etc/cron

DESCRIPTION

Cron executes commands at specified dates and times according to the instructions in the file /usr/lib/crontab. Since *cron* never exits, it should only be executed once. This is best done by running *cron* from the initialization process through the file /etc/rc; see *init*(8).

Crontab consists of lines of six fields each. The fields are separated by spaces or tabs. The first five are integer patterns to specify the minute (0-59), hour (0-23), day of the month (1-31), month of the year (1-12), and day of the week (1-7 with 1=Monday). Each of these patterns may contain a number in the range above; two numbers separated by a minus meaning a range inclusive; a list of numbers separated by commas meaning any of the numbers; or an asterisk meaning all legal values. The sixth field is a string that is executed by the Shell at the specified times. A percent character in this field is translated to a new-line character. Only the first line (up to a % or end of line) of the command field is executed by the Shell. The other lines are made available to the command as standard input.

Crontab is examined by *cron* every minute.

FILES

/usr/lib/crontab

STATUS

CRON(8) is supported by Digital Equipment Corporation.

NAME

dcheck – file system directory consistency check

SYNTAX

/etc/dcheck [**–i** numbers] [filesystem]

DESCRIPTION

Dcheck is obsoleted for normal consistency checking by *fsck*(8).

Dcheck reads the directories in a file system and compares the link-count in each i-node with the number of directory entries by which it is referenced. If the file system is not specified, a set of default file systems is checked.

The **–i** flag is followed by a list of i-numbers; when one of those i-numbers turns up in a directory, the number, the i-number of the directory, and the name of the entry are reported.

The program is fastest if the raw version of the special file is used, since the i-list is read in large chunks.

FILES

Default file systems vary with installation.

SEE ALSO

fsck(8), icheck(8), fs(5), clri(8), ncheck(8)

DIAGNOSTICS

When a file turns up for which the link-count and the number of directory entries disagree, the relevant facts are reported. Allocated files which have 0 link-count and no entries are also listed. The only dangerous situation occurs when there are more entries than links; if entries are removed, so the link-count drops to 0, the remaining entries point to thin air. They should be removed. When there are more links than entries, or there is an allocated file with neither links nor entries, some disk space may be lost but the situation will not degenerate.

RESTRICTIONS

Since *dcheck* is inherently two-pass in nature, extraneous diagnostics may be produced if applied to active file systems.

STATUS

DCHECK(8) currently is not supported by Digital Equipment Corporation.

NAME

diskpart – calculate default disk partition sizes

SYNTAX

/etc/diskpart [**–p**] [**–d**] disk-type

DESCRIPTION

Diskpart is used to calculate the disk partition sizes based on the default rules used at Berkeley. If the **–p** option is supplied, tables suitable for inclusion in a device driver are produced. If the **–d** option is supplied, an entry suitable for inclusion in the disk description file */etc/disktab* is generated; c.f. *disktab*(5). Space is always left in the last partition on the disk for a bad sector forwarding table. The space reserved is one track for the replicated copies of the table and sufficient tracks to hold a pool of 126 sectors to which bad sectors are mapped. For more information, see *bad144*(8).

The disk partition sizes are based on the total amount of space on the disk as give in the table below (all values are supplied in units of 512 byte sectors). The 'c' partition is, by convention, used to access the entire physical disk, including the space reserved for the bad sector forwarding table. In normal operation, either the 'g' partition is used, or the 'd', 'e', and 'f' partitions are used. The 'g' and 'f' partitions are variable sized, occupying whatever space remains after allocation of the fixed sized partitions. If the disk is smaller than 20 Megabytes, then *diskpart* aborts with the message "disk too small, calculate by hand".

Partition	20-60 MB	61-205 MB	206-355 MB	356+ MB
a	15884	15884	15884	15884
b	10032	33440	33440	66880
d	15884	15884	15884	15884
e	unused	55936	55936	307200
h	unused	unused	291346	291346

If an unknown disk type is specified, *diskpart* will prompt for the required disk geometry information.

SEE ALSO

disktab(5), bad144(8)

RESTRICTIONS

Certain default partition sizes are based on historical artifacts (e.g. RP06), and may result in unsatisfactory layouts.

When using the –d flag, alternate disk names are not included in the output.

Does not understand how to handle drives attached to a UDA50.

STATUS

DISKPART(8) is supported by Digital Equipment Corporation.

NAME

dmesg – collect system diagnostic messages to form error log

SYNTAX

/etc/dmesg [–]

DESCRIPTION

Dmesg looks in a system buffer for recently printed diagnostic messages and prints them on the standard output. The messages are those printed by the system when device (hardware) errors occur and (occasionally) when system tables overflow non-fatally. If the – flag is given, then *dmesg* computes (incrementally) the new messages since the last time it was run and places these on the standard output. This is typically used with *cron*(8) to produce the error log */usr/adm/messages* by running the command

```
/etc/dmesg – >> /usr/adm/messages
```

every 10 minutes.

FILES

/usr/adm/messages	error log (conventional location)
/usr/adm/msgbuf	scratch file for memory of – option

RESTRICTIONS

The system error message buffer is of small finite size. As *dmesg* is run only every few minutes, not all error messages are guaranteed to be logged.

Error diagnostics generated immediately before a system crash will never get logged.

STATUS

DMESG(8) is supported by Digital Equipment Corporation.

NAME

drtest – standalone disk test program

DESCRIPTION

Drtest is a standalone program used to read a disk track by track. It was primarily intended as a test program for new standalone drivers, but has shown useful in other contexts as well, such as verifying disks and running speed tests. For example, when a disk has been formatted (by format(8)), you can check that hard errors has been taken care of by running *drtest*. No hard errors should be found, but in many cases quite a few soft ECC errors will be reported.

While *drtest* is running, the cylinder number is printed on the console for every 10th cylinder read.

EXAMPLE

A sample run of *drtest* is shown below. In this example (using a 750), *drtest* is loaded from the root file system; usually it will be loaded from the machine's console storage device. Boldface means user input. As usual, "#" and "@" may be used to edit input.

```
>>>B/3
% %
loading hk(0,0)boot
Boot
: hk(0,0)drtest
Test program for stand-alone up and hp driver

Debugging level (1=bse, 2=ecc, 3=bse+ecc)?
Enter disk name [type(adapter,unit), e.g. hp(1,3)]? hp(0,0)
Device data: #cylinders=1024, #tracks=16, #sectors=32
Testing hp(0,0), chunk size is 16384 bytes.
(chunk size is the number of bytes read per disk access)
Start ...Make sure hp(0,0) is online
 ...
(errors are reported as they occur)
 ...
(...program restarts to allow checking other disks)
(...to abort halt machine with ^P)
```

DIAGNOSTICS

The diagnostics are intended to be self explanatory. Note, however, that the device number in the diagnostic messages is identified as *typeX* instead of *type(a,u)* where *X* = a*8+u, e.g., hp(1,3) becomes hp11.

SEE ALSO

format(8), bad144(8)

STATUS

DRTEST(8) currently is not supported by Digital Equipment Corporation.

NAME

dump – incremental file system dump

SYNTAX

/etc/dump [key [*argument* ...] filesystem]

DESCRIPTION

Dump copies to magnetic tape all files changed after a certain date in the *filesystem*. The *key* specifies the date and other options about the dump. *Key* consists of characters from the set **0123456789fusdWn.**

0–9 This number is the 'dump level'. All files modified since the last date stored in the file */etc/dumpdates* for the same filesystem at lesser levels will be dumped. If no date is determined by the level, the beginning of time is assumed; thus the option **0** causes the entire filesystem to be dumped.

f Place the dump on the next *argument* file instead of the tape. If the name of the file is "–", *dump* writes to standard output.

u If the dump completes successfully, write the date of the beginning of the dump on file */etc/dumpdates*. This file records a separate date for each filesystem and each dump level. The format of */etc/dumpdates* is readable by people, consisting of one free format record per line: filesystem name, increment level and *ctime(3)* format dump date. */etc/dumpdates* may be edited to change any of the fields, if necessary.

s The size of the dump tape is specified in feet. The number of feet is taken from the next *argument*. When the specified size is reached, *dump* will wait for reels to be changed. The default tape size is 2300 feet.

d The density of the tape, expressed in BPI, is taken from the next *argument*. This is used in calculating the amount of tape used per reel. The default is 1600.

W *Dump* tells the operator what file systems need to be dumped. This information is gleaned from the files */etc/dumpdates* and */etc/fstab*. The **W** option causes *dump* to print out, for each file system in */etc/dumpdates* the most recent dump date and level, and highlights those file systems that should be dumped. If the **W** option is set, all other options are ignored, and *dump* exits immediately.

w Is like W, but prints only those filesystems which need to be dumped.

n Whenever *dump* requires operator attention, notify by means similar to a *wall*(1) all of the operators in the group "operator".

If no arguments are given, the *key* is assumed to be **9u** and a default file system is dumped to the default tape.

Dump requires operator intervention on these conditions: end of tape, end of dump, tape write error, tape open error or disk read error (if there are more than a threshold of 32). In addition to alerting all operators implied by the **n** key, *dump* interacts with the operator on *dump's* control terminal at times when *dump* can no longer proceed, or if something is grossly wrong. All questions *dump* poses **must** be answered by typing "yes" or "no", appropriately.

Since making a dump involves a lot of time and effort for full dumps, *dump* checkpoints itself at the start of each tape volume. If writing that volume fails for some reason, *dump* will, with operator permission, restart itself from the checkpoint after the old tape has been rewound and removed, and a new tape has been mounted.

Dump tells the operator what is going on at periodic intervals, including usually low estimates of the number of blocks to write, the number of tapes it will take, the time to completion, and the time to the tape change. The output is verbose, so that others know that the terminal controlling *dump* is busy, and will be for some time.

Now a short suggestion on how to perform dumps. Start with a full level 0 dump

```
dump 0un
```

Next, dumps of active file systems are taken on a daily basis, using a modified Tower of Hanoi algorithm, with this sequence of dump levels:

3 2 5 4 7 6 9 8 9 9 ...

For the daily dumps, a set of 10 tapes per dumped file system is used on a cyclical basis. Each week, a level 1 dump is taken, and the daily Hanoi sequence repeats with 3. For weekly dumps, a set of 5 tapes per dumped file system is used, also on a cyclical basis. Each month, a level 0 dump is taken on a set of fresh tapes that is saved forever.

FILES

/dev/rrp1g	default filesystem to dump from
/dev/rmt8	default tape unit to dump to
/etc/ddate	old format dump date record (obsolete after **−J** option)
/etc/dumpdates	new format dump date record
/etc/fstab	dump table: file systems and frequency
/etc/group	to find group *operator*

SEE ALSO

restore(8), dump(5), fstab(5)

RESTRICTIONS

Sizes are based on 1600 BPI blocked tape; the raw magtape device has to be used to approach these densities. Fewer than 32 read errors on the filesystem are ignored. Each reel requires a new process, so parent processes for reels already written just hang around until the entire tape is written.

It would be nice if *dump* knew about the dump sequence, kept track of the tapes scribbled on, told the operator which tape to mount when, and provided more assistance for the operator running *restore*.

STATUS

DUMP(8) is supported by Digital Equipment Corporation.

NAME

dumpfs – dump file system information

SYNTAX

dumpfs *filesys|device*

DESCRIPTION

Dumpfs prints out the super block and cylinder group information for the file system or special device specified. The listing is very long and detailed. This command is useful mostly for finding out certain file system information such as the file system block size and minimum free space percentage.

SEE ALSO

fs(5), disktab(5), tunefs(8), newfs(8), fsck(8)

STATUS

DUMPFS(8) currently is not supported by Digital Equipment Corporation.

NAME

edquota – edit user quotas

SYNTAX

edquota [**–p** *proto-user*] *users...*

DESCRIPTION

Edquota is a quota editor. One or more users may be specified on the command line. For each user a temporary file is created with an ASCII representation of the current disc quotas for that user and an editor is then invoked on the file. The quotas may then be modified, new quotas added, etc. Upon leaving the editor, *edquota* reads the temporary file and modifies the binary quota files to reflect the changes made.

If the **–p** option is specified, *edquota* will duplicate the quotas of the prototypical user specified for each user specified. This is the normal mechanism used to initialize quotas for groups of users.

The editor invoked is *vi*(1) unless the environment variable EDITOR specifies otherwise.

Only the super-user may edit quotas.

FILES

quotas	at the root of each file system with quotas
/etc/fstab	to find file system names and locations

SEE ALSO

quota(1), quota(2), quotacheck(8), quotaon(8), repquota(8)

DIAGNOSTICS

Various messages about inaccessible files; self-explanatory.

STATUS

EDQUOTA(8) currently is not supported by Digital Equipment Corporation.

NAME

fastboot, fasthalt – reboot/halt the system without checking the disks

SYNTAX

/etc/fastboot [*boot-options*]
/etc/fasthalt [*halt-options*]

DESCRIPTION

Fastboot and *fasthalt* are shell scripts which reboot and halt the system without checking the file systems. This is done by creating a file */fastboot*, then invoking the *reboot* program. The system startup script, */etc/rc*, looks for this file and, if present, skips the normal invocation of *fsck*(8).

SEE ALSO

halt(8), reboot(8), rc(8)

STATUS

FASTBOOT(8) currently is not supported by Digital Equipment Corporation.

NAME

format – how to format disk packs

DESCRIPTION

There are two ways to format disk packs. The simplest is to use the *format* program. The alternative is to use the DEC standard formatting software which operates under the DEC diagnostic supervisor. This manual page describes the operation of *format*, then concludes with some remarks about using the DEC formatter.

Format is a standalone program used to format and check disks prior to constructing file systems. In addition to the formatting operation, *format* records any bad sectors encountered according to DEC standard 144. Formatting is performed one track at a time by writing the appropriate headers and a test pattern and then checking the sector by reading and verifying the pattern, using the controller's ECC for error detection. A sector is marked bad if an unrecoverable media error is detected, or if a correctable ECC error greater than 5 bits in length is detected (such errors are indicated as "ECC" in the summary printed upon completing the format operation). After the entire disk has been formatted and checked, the total number of errors are reported, any bad sectors and skip sectors are marked, and a bad sector forwarding table is written to the disk in the first five even numbered sectors of the last track. *Format* may be used on any UNIBUS or MASSBUS drive supported by the *up* and *hp* device drivers which uses 4-byte headers (everything except RP's).

The test pattern used during the media check may be selected from one of: 0xf00f (RH750 worst case), 0xec6d (media worst case), and 0xa5a5 (alternating 1's and 0's). Normally the media worst case pattern is used.

Format also has an option to perform an extended "severe burnin," which makes 46 passes using different patterns. Using this option, sectors with any errors of any size are marked bad. This test runs for many hours, depending on the disk and processor.

Each time *format* is run a completely new bad sector table is generated based on errors encountered while formatting. The device driver, however, will always attempt to read any existing bad sector table when the device is first opened. Thus, if a disk pack has never previously been formatted, or has been formatted with different sectoring, five error messages will be printed when the driver attempts to read the bad sector table; these diagnostics should be ignored.

Formatting a 400 megabyte disk on a MASSBUS disk controller usually takes about 20 minutes. Formatting on a UNIBUS disk controller takes significantly longer. For every hundredth cylinder formatted *format* prints a message indicating the current cylinder being formatted. (This message is just to reassure people that nothing is is amiss.)

Format uses the standard notation of the standalone i/o library in identifying a drive to be formatted. A drive is specified as *zz*(*x*,*y*), where *zz* refers to the controller type (either *hp* or *up*), *x* is the unit number of the drive; 8 times the UNIBUS or MASSBUS adaptor number plus the MASSBUS drive number or UNIBUS drive unit number; and *y* is the file system partition on drive *x* (this should always be 0). For example, "hp(1,0)" indicates that drive 1 on MASSBUS adaptor 0 should be formatted; while "up(10,0)" indicates UNIBUS drive 2 on UNIBUS adaptor 1 should be formatted.

Before each formatting attempt, *format* prompts the user in case debugging should be enabled in the appropriate device driver. A carriage return disables debugging information.

Format should be used prior to building file systems (with *newfs*(8)) to insure all sectors with uncorrectable media errors are remapped. If a drive develops uncorrectable defects after formatting, the program *badsect*(8) must be used.

EXAMPLE

A sample run of *format* is shown below. In this example (using a VAX-11/780), *format* is loaded from the console floppy; on an 11/750 *format* will be loaded from the root file system. Boldface means user input. As usual, "#" and "@" may be used to edit input.

```
>>>L FORMAT
                    LOAD DONE, 00004400 BYTES LOADED
>>>S 2
Disk format/check utility

Enable debugging (0=none, 1=bse, 2=ecc, 3=bse+ecc)? 0
Device to format? hp(8,0)
(error messages may occur as old bad sector table is read)
Formatting drive hp0 on adaptor 1: verify (yes/no)? yes
Device data: #cylinders=842, #tracks=20, #sectors=48
Available test patterns are:
                    1 - (f00f) rh750 worst case
                    2 - (ec6d) media worst case
                    3 - (a5a5) alternating 1's and 0's
                    4 - (ffff) Severe burnin (takes several hours)
Pattern (one of the above, other to restart)? 2
Start formatting...make sure the drive is online
 ...
(soft ecc's and other errors are reported as they occur)
 ...
(if 4 write check errors were found, the program terminates like this...)
 ...
Errors:
Write check: 4
Bad sector: 0
ECC: 0
Skip sector: 0
Total of 4 hard errors found.
Writing bad sector table at block 808271
(808271 is the block # of the first block in the bad sector table)
Done
(...program restarts to allow formatting other disks)
(...to abort halt machine with ^P)
```

DIAGNOSTICS

The diagnostics are intended to be self explanatory.

USING DEC SOFTWARE TO FORMAT

Warning: These instructions are for people with 11/780 CPU's. The steps needed for 11/750 or 11/730 cpu's are similar, but not covered in detail here.

The formatting procedures are different for each type of disk. Listed here are the formatting procedures for RK07's, RP0X, and RM0X disks.

You should shut down UNIX and halt the machine to do any disk formatting. Make certain you put in the pack you want formatted. It is also a good idea to spin down or write protect the disks you don't want to format, just in case.

Formatting an RK07. Load the console floppy labeled, "RX11 VAX DSK LD DEV #1" in the console disk drive, and type the following commands:

```
>>>BOOT
DIAGNOSTIC SUPERVISOR. ZZ-ESSAA-X5.0-119  23-JAN-1980 12:44:40.03
DS>ATTACH DW780 SBI DW0 3 5
DS>ATTACH RK611 DMA
DS>ATTACH RK07 DW0 DMA0
DS>SELECT DMA0
DS>LOAD EVRAC
DS>START/SEC:PACKINIT
```

Formatting an RP0X. Follow the above procedures except that the ATTACH and SELECT lines should read:

```
DS>ATTACH RH780 SBI RH0 8 5
DS>ATTACH RP0X RH0 DBA0(RP0X is, e.g. RP06)
DS>SELECT DBA0
```

This is for drive 0 on mba0; use 9 instead of 8 for mba1, etc.

Formatting an RM0X. Follow the above procedures except that the ATTACH and SELECT lines should read:

```
DS>ATTACH RH780 SBI RH0 8 5
DS>ATTACH RM0X RH0 DRA0
DS>SELECT DRA0
```

Don't forget to put your UNIX console floppy back in the floppy disk drive.

SEE ALSO

bad144(8), badsect(8), newfs(8)

STATUS

FORMAT(8V) currently is not supported by Digital Equipment Corporation.

NAME

fsck – file system consistency check and interactive repair

SYNTAX

/etc/fsck **–p** [filesystem ...]
/etc/fsck [**–b** block#] [**–y**] [**–n**] [filesystem] ...

DESCRIPTION

The first form of *fsck* preens a standard set of filesystems or the specified file systems. It is normally used in the script **/etc/rc** during automatic reboot. In this case *fsck* reads the table **/etc/fstab** to determine which file systems to check. It uses the information there to inspect groups of disks in parallel taking maximum advantage of i/o overlap to check the file systems as quickly as possible. Normally, the root file system will be checked on pass 1, other "root" ("a" partition) file systems on pass 2, other small file systems on separate passes (e.g. the "d" file systems on pass 3 and the "e" file systems on pass 4), and finally the large user file systems on the last pass, e.g. pass 5. A pass number of 0 in fstab causes a disk to not be checked; similarly partitions which are not shown as to be mounted "rw" or "ro" are not checked.

The system takes care that only a restricted class of innocuous inconsistencies can happen unless hardware or software failures intervene. These are limited to the following:

Unreferenced inodes

Link counts in inodes too large

Missing blocks in the free list

Blocks in the free list also in files

Counts in the super-block wrong

These are the only inconsistencies which *fsck* with the **–p** option will correct; if it encounters other inconsistencies, it exits with an abnormal return status and an automatic reboot will then fail. For each corrected inconsistency one or more lines will be printed identifying the file system on which the correction will take place, and the nature of the correction. After successfully correcting a file system, *fsck* will print the number of files on that file system and the number of used and free blocks.

Without the **–p** option, *fsck* audits and interactively repairs inconsistent conditions for file systems. If the file system is inconsistent the operator is prompted for concurrence before each correction is attempted. It should be noted that a number of the corrective actions which are not fixable under the **–p** option will result in some loss of data. The amount and severity of data lost may be determined from the diagnostic output. The default action for each consistency correction is to wait for the operator to respond **yes** or **no**. If the operator does not have write permission *fsck* will default to a **–n** action.

Fsck has more consistency checks than its predecessors *check, dcheck, fcheck,* and *icheck* combined.

The following flags are interpreted by *fsck*.

-b Use the block specified immediately after the flag as the super block for the file system. Block 32 is always an alternate super block.

-y Assume a yes response to all questions asked by *fsck;* this should be used with great caution as this is a free license to continue after essentially unlimited trouble has been encountered.

-n Assume a no response to all questions asked by *fsck;* do not open the file system for writing.

If no filesystems are given to *fsck* then a default list of file systems is read from the file **/etc/fstab**.

Inconsistencies checked are as follows:

1. Blocks claimed by more than one inode or the free list.
2. Blocks claimed by an inode or the free list outside the range of the file system.
3. Incorrect link counts.
4. Size checks:
   Directory size not of proper format.
5. Bad inode format.
6. Blocks not accounted for anywhere.
7. Directory checks:
   File pointing to unallocated inode.
   Inode number out of range.
8. Super Block checks:

   More blocks for inodes than there are in the file system.
9. Bad free block list format.
10. Total free block and/or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the **lost+found** directory. The name assigned is the inode number. The only restriction is that the directory **lost+found** must preexist in the root of the filesystem being checked and must have empty slots in which entries can be made. This is accomplished by making **lost+found**, copying a number of files to the directory, and then removing them (before *fsck* is executed).

Checking the raw device is almost always faster.

FILES

/etc/fstab contains default list of file systems to check.

DIAGNOSTICS

The diagnostics produced by *fsck* are intended to be self-explanatory.

SEE ALSO

fstab(5), fs(5), newfs(8), mkfs(8), crash(8V), reboot(8)

RESTRICTIONS

Inode numbers for . and .. in each directory should be checked for validity.

STATUS

FSCK(8) is supported by Digital Equipment Corporation.

NAME

ftpd – DARPA Internet File Transfer Protocol server

SYNTAX

/etc/ftpd [**–d**] [**–l**] [**–t**timeout]

DESCRIPTION

Ftpd is the DARPA Internet File Transfer Prototocol server process. The server uses the TCP protocol and listens at the port specified in the "ftp" service specification; see *services*(5).

If the **–d** option is specified, each socket created will have debugging turned on (SO DEBUG). With debugging enabled, the system will trace all TCP packets sent and received on a socket. The program *trpt*(8C) may then be used to interpret the packet traces.

If the **–l** option is specified, each ftp session is logged on the standard output. This allows a line of the form '/etc/ftpd -l > /tmp/ftplog" to be used to conveniently maintain a log of ftp sessions.

The ftp server will timeout an inactive session after 60 seconds. If the **–t** option is specified, the inactivity timeout period will be set to *timeout*.

The ftp server currently supports the following ftp requests; case is not distinguished.

Request	**Description**
ACCT	specify account (ignored)
ALLO	allocate storage (vacuously)
APPE	append to a file
CWD	change working directory
DELE	delete a file
HELP	give help information
LIST	give list files in a directory ("ls -lg")
MODE	specify data transfer *mode*
NLST	give name list of files in directory ("ls")
NOOP	do nothing
PASS	specify password
PORT	specify data connection port
QUIT	terminate session
RETR	retrieve a file
RNFR	specify rename-from file name
RNTO	specify rename-to file name
STOR	store a file
STRU	specify data transfer *structure*
TYPE	specify data transfer *type*
USER	specify user name
XCUP	change to parent of current working directory
XCWD	change working directory
XMKD	make a directory

XPWD print the current working directory
XRMD remove a directory

The remaining ftp requests specified in Internet RFC 765 are recognized, but not implemented.

Ftpd interprets file names according to the "globbing" conventions used by *csh*(1). This allows users to utilize the metacharacters "*?[]{}~".

Ftpd authenticates users according to three rules.

1) The user name must be in the password data base, */etc/passwd*, and not have a null password. In this case a password must be provided by the client before any file operations may be performed.

2) The user name must not appear in the file */etc/ftpusers*.

3) If the user name is "anonymous" or "ftp", an anonymous ftp account must be present in the password file (user "ftp"). In this case the user is allowed to log in by specifying any password (by convention this is given as the client host's name).

In the last case, *ftpd* takes special measures to restrict the client's access privileges. The server performs a *chroot*(2) command to the home directory of the "ftp" user. In order that system security is not breached, it is recommended that the "ftp" subtree be constructed with care; the following rules are recommended.

~ftp) Make the home directory owned by "ftp" and unwritable by anyone.

~ftp/bin)
Make this directory owned by the super-user and unwritable by anyone. The program *ls*(1) must be present to support the list commands. This program should have mode 111.

~ftp/etc)
Make this directory owned by the super-user and unwritable by anyone. The files *passwd*(5) and *group*(5) must be present for the *ls* command to work properly. These files should be mode 444.

~ftp/pub)
Make this directory mode 777 and owned by "ftp". Users should then place files which are to be accessible via the anonymous account in this directory.

SEE ALSO

ftp(1C),

RESTRICTIONS

There is no support for aborting commands.

The anonymous account is inherently dangerous and should avoided when possible.

The server must run as the super-user to create sockets with privileged port numbers. It maintains an effective user id of the logged in user, reverting to the super-user only when binding addresses to sockets. The possible security holes have been extensively scrutinized,

but are possibly incomplete.

STATUS

FTPD(8C) currently is not supported by Digital Equipment Corporation.

NAME

gettable – get NIC format host tables from a host

SYNTAX

/etc/gettable *host*

DESCRIPTION

Gettable is a simple program used to obtain the NIC standard host tables from a "nicname" server. The indicated *host* is queried for the tables. The tables, if retrieved, are placed in the file *hosts.txt*.

Gettable operates by opening a TCP connection to the port indicated in the service specification for "nicname". A request is then made for "ALL" names and the resultant information is placed in the output file.

Gettable is best used in conjunction with the *htable*(8) program which converts the NIC standard file format to that used by the network library lookup routines.

SEE ALSO

intro(3N), htable(8)

STATUS

GETTABLE(8C) currently is not supported by Digital Equipment Corporation.

NAME

getty – set terminal mode

SYNTAX

/etc/getty [type]

DESCRIPTION

Getty is invoked by *init*(8) immediately after a terminal is opened, following the making of a connection. While reading the name *getty* attempts to adapt the system to the speed and type of terminal being used.

Init calls *getty* with an argument specified by the *ttys* file entry for the terminal line. The argument can be used to make *getty* treat the line specially. This argument is used as an index into the *gettytab*(5) database, to determine the characteristics of the line. If there is no argument, or there is no such table, the **default** table is used. If there is no **/etc/gettytab** a set of system defaults is used. If indicated by the table located, *getty* will clear the terminal screen, print a banner heading, and prompt for a login name. Usually either the banner of the login prompt will include the system hostname. Then the user's name is read, a character at a time. If a null character is received, it is assumed to be the result of the user pushing the 'break' ('interrupt') key. The speed is usually then changed and the 'login:' is typed again; a second 'break' changes the speed again and the 'login:' is typed once more. Successive 'break' characters cycle through the some standard set of speeds.

The user's name is terminated by a new-line or carriage-return character. The latter results in the system being set to treat carriage returns appropriately (see *tty*(4)).

The user's name is scanned to see if it contains any lower-case alphabetic characters; if not, and if the name is nonempty, the system is told to map any future upper-case characters into the corresponding lower-case characters.

Finally, login is called with the user's name as argument.

Most of the default actions of *getty* can be circumvented, or modified, by a suitable *gettytab* table.

Getty can be set to timeout after some interval, which will cause dial up lines to hang up if the login name is not entered reasonably quickly.

FILES

/etc/gettytab

SEE ALSO

gettytab(5), init(8), login(1), ioctl(2), tty(4), ttys(5).

RESTRICTIONS

Currently, the format of **/etc/ttys** limits the permitted table names to a single character.

STATUS

GETTY(8) is supported by Digital Equipment Corporation.

NAME

halt – stop the processor

SYNTAX

/etc/halt [**–n**] [**–q**] [**–y**]

DESCRIPTION

Halt writes out sandbagged information to the disks and then stops the processor. The machine does not reboot, even if the auto-reboot switch is set on the console.

The **–n** option prevents the sync before stopping. The **–q** option causes a quick halt, no graceful shutdown is attempted. The **–y** option is needed if you are trying to halt the system from a dialup.

SEE ALSO

reboot(8), shutdown(8)

RESTRICTIONS

It is very difficult to halt a VAX, as the machine wants to then reboot itself. A rather tight loop suffices.

STATUS

HALT(8) is supported by Digital Equipment Corporation.

NAME

htable – convert NIC standard format host tables

SYNTAX

/etc/htable *file*

DESCRIPTION

Htable is used to convert host files in the format specified in Internet RFC 810 to the format used by the network library routines. Three files are created as a result of running *htable*: *hosts*, *networks*, and *gateways*. The *hosts* file is used by the *gethostent*(3N) routines in mapping host names to addresses. The *networks* file is used by the *getnetent*(3N) routines in mapping network names to numbers. The *gateways* file is used by the routing daemon in identifying "passive" Internet gateways; see *routed*(8C) for an explanation.

If any of the files *localhosts*, *localnetworks*, or *localgateways* are present in the current directory, the file's contents is prepended to the output file without interpretation. This allows sites to maintain local aliases and entries which are not normally present in the master database.

Htable is best used in conjunction with the *gettable*(8C) program which retrieves the NIC database from a host.

SEE ALSO

intro(3N), gettable(8C)

RESTRICTIONS

Does not properly calculate the *gateways* file.

STATUS

HTABLE(8) currently is not supported by Digital Equipment Corporation.

NAME

icheck – file system storage consistency check

SYNTAX

/etc/icheck [**–s**] [**–b** numbers] [filesystem]

DESCRIPTION

Icheck is obsoleted for normal consistency checking by *fsck*(8).

Icheck examines a file system, builds a bit map of used blocks, and compares this bit map against the free list maintained on the file system. If the file system is not specified, a set of default file systems is checked. The normal output of *icheck* includes a report of

The total number of files and the numbers of regular, directory, block special and character special files.

The total number of blocks in use and the numbers of single-, double-, and triple-indirect blocks and directory blocks.

The number of free blocks.

The number of blocks missing; i.e. not in any file nor in the free list.

The **–s** option causes *icheck* to ignore the actual free list and reconstruct a new one by rewriting the super-block of the file system. The file system should be dismounted while this is done; if this is not possible (for example if the root file system has to be salvaged) care should be taken that the system is quiescent and that it is rebooted immediately afterwards so that the old, bad in-core copy of the super-block will not continue to be used. Notice also that the words in the super-block which indicate the size of the free list and of the i-list are believed. If the super-block has been curdled these words will have to be patched. The **–s** option causes the normal output reports to be suppressed.

Following the **–b** option is a list of block numbers; whenever any of the named blocks turns up in a file, a diagnostic is produced.

Icheck is faster if the raw version of the special file is used, since it reads the i-list many blocks at a time.

FILES

Default file systems vary with installation.

SEE ALSO

fsck(8), dcheck(8), ncheck(8), fs(5), clri(8)

DIAGNOSTICS

For duplicate blocks and bad blocks (which lie outside the file system) *icheck* announces the difficulty, the i-number, and the kind of block involved. If a read error is encountered, the block number of the bad block is printed and *icheck* considers it to contain 0. 'Bad freeblock' means that a block number outside the available space was encountered in the free list. '*n* dups in free' means that *n* blocks were found in the free list which duplicate blocks either in some file or in the earlier part of the free list.

RESTRICTIONS

Since *icheck* is inherently two-pass in nature, extraneous diagnostics may be produced if applied to active file systems.

It believes even preposterous super-blocks and consequently can get core images.

STATUS

ICHECK(8) currently is not supported by Digital Equipment Corporation.

NAME

implog – IMP log interpreter

SYNTAX

/etc/implog [**–D**] [**–f**] [**–c**] [**–l** [*link*]] [**–h** *host#*] [**–i** *imp#*] [**–t** *message-type*]

DESCRIPTION

Implog is program which interprets the message log produced by *implogd*(8C).

If no arguments are specified, *implog* interprets and prints every message present in the message file. Options may be specified to force printing only a subset of the logged messages.

–D Do not show data messages.

–f Follow the logging process in action. This flags causes *implog* to print the current contents of the log file, then check for new logged messages every 5 seconds.

–c In addition to printing any data messages logged, show the contents of the data in hexadecimal bytes.

–l [*link#*]
Show only those messages received on the specified "link". If no value is given for the link, the link number of the IP protocol is assumed.

–h *host#*
Show only those messages received from the specified host. (Usually specified in conjunction with an imp.)

–i *imp#*
Show only those messages received from the specified imp.

–t *message-type*
Show only those messages received of the specified message type.

SEE ALSO

imp(4P), implogd(8C)

RESTRICTIONS

Can not specify multiple hosts, imps, etc. Can not follow reception of messages without looking at those currently in the file.

STATUS

IMPLOG(8C) currently is not supported by Digital Equipment Corporation.

NAME

implogd – IMP logger process

SYNTAX

/etc/implogd [–d]

DESCRIPTION

Implogd is program which logs messages from the IMP, placing them in the file */usr/adm/implog*.

Entries in the file are variable length. Each log entry has a fixed length header of the form:

```
struct sockstamp {
        short   sin_family;
        u_short sin_port;
        struct  in_addr sin_addr;
        time_t  sin_time;
        int     sin_len;
};
```

followed, possibly, by the message received from the IMP. Each time the logging process is started up it places a time stamp entry in the file (a header with *sin len* field set to 0).

The logging process will catch only those message from the IMP which are not processed by a protocol module, e.g. IP. This implies the log should contain only status information such as "IMP going down" messages and, perhaps, stray NCP messages.

SEE ALSO

imp(4P), implog(8C)

STATUS

IMPLOGD(8C) currently is not supported by Digital Equipment Corporation.

NAME

init – process control initialization

SYNTAX

/etc/init

DESCRIPTION

Init is invoked inside UNIX as the last step in the boot procedure. It normally then runs the automatic reboot sequence as described in *reboot*(8), and if this succeeds, begins multi-user operation. If the reboot fails, it commences single user operation by giving the super-user a shell on the console. It is possible to pass parameters from the boot program to *init* so that single user operation is commenced immediately. When such single user operation is terminated by killing the single-user shell (i.e. by hitting ^D), *init* runs */etc/rc* without the reboot parameter. This command file performs housekeeping operations such as removing temporary files, mounting file systems, and starting daemons.

In multi-user operation, *init's* role is to create a process for each terminal port on which a user may log in. To begin such operations, it reads the file */etc/ttys* and forks several times to create a process for each terminal specified in the file. Each of these processes opens the appropriate terminal for reading and writing. These channels thus receive file descriptors 0, 1 and 2, the standard input and output and the diagnostic output. Opening the terminal will usually involve a delay, since the *open* is not completed until someone is dialed up and carrier established on the channel. If a terminal exists but an error occurs when trying to open the terminal *init* complains by writing a message to the system console; the message is repeated every 10 minutes for each such terminal until the terminal is shut off in /etc/ttys and init notified (by a hangup, as described below), or the terminal becomes accessible (init checks again every minute). After an open succeeds, */etc/getty* is called with argument as specified by the second character of the *ttys* file line. *Getty* reads the user's name and invokes *login* to log in the user and execute the Shell.

Ultimately the Shell will terminate because of an end-of-file either typed explicitly or generated as a result of hanging up. The main path of *init*, which has been waiting for such an event, wakes up and removes the appropriate entry from the file *utmp*, which records current users, and makes an entry in */usr/adm/wtmp*, which maintains a history of logins and logouts. The *wtmp* entry is made only if a user logged in successfully on the line. Then the appropriate terminal is reopened and *getty* is reinvoked.

Init catches the *hangup* signal (signal SIGHUP) and interprets it to mean that the file */etc/ttys* should be read again. The Shell process on each line which used to be active in *ttys* but is no longer there is terminated; a new process is created for each added line; lines unchanged in the file are undisturbed. Thus it is possible to drop or add phone lines without rebooting the system by changing the *ttys* file and sending a *hangup* signal to the *init* process: use 'kill –HUP 1.'

Init will terminate multi-user operations and resume single-user mode if sent a terminate (TERM) signal, i.e. "kill –TERM 1". If there are processes outstanding which are deadlocked (due to hardware or software failure), *init* will not wait for them all to die (which might take forever), but will time out after 30 seconds and print a warning message.

Init will cease creating new *getty*'s and allow the system to slowly die away, if it is sent a terminal stop (TSTP) signal, i.e. "kill −TSTP 1". A later hangup will resume full multi-user operations, or a terminate will initiate a single user shell. This hook is used by *reboot*(8) and *halt*(8).

Init's role is so critical that if it dies, the system will reboot itself automatically. If, at bootstrap time, the *init* process cannot be located, the system will loop in user mode at location 0x13.

DIAGNOSTICS

init: *tty*: **cannot open**. A terminal which is turned on in the *rc* file cannot be opened, likely because the requisite lines are either not configured into the system or the associated device was not attached during boot-time system configuration.

WARNING: Something is hung (wont die); ps axl advised. A process is hung and could not be killed when the system was shutting down. This is usually caused by a process which is stuck in a device driver due to a persistent device error condition.

FILES

/dev/console, /dev/tty*, /etc/utmp, /usr/adm/wtmp, /etc/ttys, /etc/rc

SEE ALSO

login(1), kill(1), sh(1), ttys(5), crash(8V), getty(8), rc(8), reboot(8), halt(8), shutdown(8)

STATUS

INIT(8) is supported by Digital Equipment Corporation.

NAME

kgmon – generate a dump of the operating system's profile buffers

SYNTAX

/etc/kgmon [**–b**] [**–h**] [**–r**] [**–p**] [system] [memory]

DESCRIPTION

Kgmon is a tool used when profiling the operating system. When no arguments are supplied, *kgmon* indicates the state of operating system profiling as running, off, or not configured. (see *config*(8)) If the **–p** flag is specified, *kgmon* extracts profile data from the operating system and produces a *gmon.out* file suitable for later analysis by *gprof*(1).

The following options may be specified:

–b Resume the collection of profile data.

–h Stop the collection of profile data.

–p Dump the contents of the profile buffers into a *gmon.out* file.

–r Reset all the profile buffers. If the **–p** flag is also specified, the *gmon.out* file is generated before the buffers are reset.

If neither **–b** nor **–h** is specified, the state of profiling collection remains unchanged. For example, if the **–p** flag is specified and profile data is being collected, profiling will be momentarily suspended, the operating system profile buffers will be dumped, and profiling will be immediately resumed.

FILES

/vmunix – the default system
/dev/kmem – the default memory

SEE ALSO

gprof(1), config(8)

DIAGNOSTICS

Users with only read permission on /dev/kmem cannot change the state of profiling collection. They can get a *gmon.out* file with the warning that the data may be inconsistent if profiling is in progress.

STATUS

KGMON(8) currently is not supported by Digital Equipment Corporation.

NAME

lpc – line printer control program

SYNTAX

/etc/lpc [command [argument ...]]

DESCRIPTION

Lpc is used by the system administrator to control the operation of the line printer system. For each line printer configured in /etc/printcap, *lpc* may be used to:

- disable or enable a printer,
- disable or enable a printer's spooling queue,
- rearrange the order of jobs in a spooling queue,
- find the status of printers, and their associated spooling queues and printer dameons.

Without any arguments, *lpc* will prompt for commands from the standard input. If arguments are supplied, *lpc* interprets the first argument as a command and the remaining arguments as parameters to the command. The standard input may be redirected causing *lpc* to read commands from file. Commands may be abreviated; the following is the list of recognized commands.

? [command ...]

help [command ...]
: Print a short description of each command specified in the argument list, or, if no arguments are given, a list of the recognized commands.

abort { all | printer ... }
: Terminate an active spooling daemon on the local host immediately and then disable printing (preventing new daemons from being started by *lpr*) for the specified printers.

clean { all | printer ... }
: Remove all files beginning with "cf", "tf", or "df" from the specified printer queue(s) on the local machine.

enable { all | printer ... }
: Enable spooling on the local queue for the listed printers. This will allow *lpr* to put new jobs in the spool queue.

exit

quit
: Exit from lpc.

disable { all | printer ... }
: Turn the specified printer queues off. This prevents new printer jobs from being entered into the queue by *lpr*.

restart { all | printer ... }

Attempt to start a new printer daemon. This is useful when some abnormal condition causes the daemon to die unexpectedly leaving jobs in the queue. *Lpq* will report that there is no daemon present when this condition occurs.

start { all | printer ... }
Enable printing and start a spooling daemon for the listed printers.

status [all] [printer ...]
Display the status of daemons and queues on the local machine.

stop { all | printer ... }
Stop a spooling daemon after the current job completes and disable printing.

topq printer [jobnum ...] [user ...]
Place the jobs in the order listed at the top of the printer queue.

FILES

/etc/printcap	printer description file
/usr/spool/*	spool directories
/usr/spool/*/lock	lock file for queue control

SEE ALSO

lpd(8), lpr(1), lpq(1), lprm(1), printcap(5)

DIAGNOSTICS

?Ambiguous command	abreviation matches more than one command
?Invalid command	no match was found
?Privileged command	command can be executed by root only

STATUS

LPC(8) is supported by Digital Equipment Corporation.

NAME

lpd – line printer daemon

SYNTAX

/usr/lib/lpd [-l] [-L logfile] [port #]

DESCRIPTION

Lpd is the line printer daemon (spool area handler) and is normally invoked at boot time from the *rc*(8) file. It makes a single pass through the *printcap*(5) file to find out about the existing printers and prints any files left after a crash. It then uses the system calls *listen*(2) and *accept*(2) to receive requests to print files in the queue, transfer files to the spooling area, display the queue, or remove jobs from the queue. In each case, it forks a child to handle the request so the parent can continue to listen for more requests. The Internet port number used to rendezvous with other processes is normally obtained with *getservbyname*(3) but can be changed with the *port#* argument. The **–L** option changes the file used for writing error conditions from the system console to *logfile*. The **–l** flag causes *lpd* to log valid requests received from the network. This can be useful for debugging purposes.

Access control is provided by two means. First, All requests must come from one of the machines listed in the file */etc/hosts.equiv*. Second, if the "rs" capability is specified in the *printcap* entry for the printer being accessed, *lpr* requests will only be honored for those users with accounts on the machine with the printer.

The file *lock* in each spool directory is used to prevent multiple daemons from becoming active simultaneously, and to store information about the daemon process for *lpr*(1), *lpq*(1), and *lprm*(1). After the daemon has successfully set the lock, it scans the directory for files beginning with *cf*. Lines in each *cf* file specify files to be printed or non-printing actions to be performed. Each such line begins with a key character to specify what to do with the remainder of the line.

J Job Name. String to be used for the job name on the burst page.

C Classification. String to be used for the classification line on the burst page.

L Literal. The line contains identification info from the password file and causes the banner page to be printed.

T Title. String to be used as the title for *pr*(1).

H Host Name. Name of the machine where *lpr* was invoked.

P Person. Login name of the person who invoked *lpr*. This is used to verify ownership by *lprm*.

M Send mail to the specified user when the current print job completes.

f Formatted File. Name of a file to print which is already formatted.

l Like "f" but passes control characters and does not make page breaks.

p Name of a file to print using *pr*(1) as a filter.

t Troff File. The file contains *troff*(1) output (cat phototypesetter commands).

d DVI File. The file contains *Tex*(l) output (DVI format from Standford).

g Graph File. The file contains data produced by *plot*(3X).

c Cifplot File. The file contains data produced by *cifplot*.

v The file contains a raster image.

r The file contains text data with FORTRAN carriage control characters.

1 Troff Font R. Name of the font file to use instead of the default.

2 Troff Font I. Name of the font file to use instead of the default.

3 Troff Font B. Name of the font file to use instead of the default.

4 Troff Font S. Name of the font file to use instead of the default.

W Width. Changes the page width (in characters) used by *pr*(1) and the text filters.

I Indent. The number of characters to indent the output by (in ascii).

U Unlink. Name of file to remove upon completion of printing.

N File name. The name of the file which is being printed, or a blank for the standard input (when *lpr* is invoked in a pipeline).

If a file can not be opened, a message will be placed in the log file (normally the console). *Lpd* will try up to 20 times to reopen a file it expects to be there, after which it will skip the file to be printed.

Lpd uses *flock*(2) to provide exclusive access to the lock file and to prevent multiple deamons from becoming active simultaneously. If the daemon should be killed or die unexpectedly, the lock file need not be removed. The lock file is kept in a readable ASCII form and contains two lines. The first is the process id of the daemon and the second is the control file name of the current job being printed. The second line is updated to reflect the current status of *lpd* for the programs *lpq*(1) and *lprm*(1).

FILES

/etc/printcap	printer description file
/usr/spool/*	spool directories
/dev/lp*	line printer devices
/dev/printer	socket for local requests
/etc/hosts.equiv	lists machine names allowed printer access

SEE ALSO

lpc(8), pac(1), lpr(1), lpq(1), lprm(1), printcap(5)
4.2BSD Line Printer Spooler Manual

STATUS

LPD(8) is supported by Digital Equipment Corporation.

NAME

makedev – make system special files

SYNTAX

/dev/MAKEDEV *device...*

DESCRIPTION

MAKEDEV is a shell script normally used to install special files. It resides in the */dev* directory, as this is the normal location of special files. Arguments to *MAKEDEV* are usually of the form *device-name*? where *device-name* is one of the supported devices listed in section 4 of the manual and "?" is a logical unit number (0-9). A few special arguments create assorted collections of devices and are listed below.

std Create the *standard* devices for the system; e.g. /dev/console, /dev/tty. The VAX-11/780 console floppy device, /dev/floppy, and VAX-11/750 and VAX-11/730 console cassette device(s), /dev/tu?, are also created with this entry.

local Create those devices specific to the local site. This request causes the shell file */dev/MAKEDEV.local* to be executed. Site specific commands, such as those used to setup dialup lines as "ttyd?" should be included in this file.

Since all devices are created using *mknod*(8), this shell script is useful only to the superuser.

DIAGNOSTICS

Either self-explanatory, or generated by one of the programs called from the script. Use "sh -x MAKEDEV" in case of trouble.

SEE ALSO

intro(4), config(8), mknod(8)

RESTRICTIONS

When more than one piece of hardware of the same "kind" is present on a machine (for instance, a dh and a dmf), naming conflicts arise.

STATUS

MAKEDEV(8) currently is not supported by Digital Equipment Corporation.

NAME

makekey – generate encryption key

SYNTAX

/usr/lib/makekey

DESCRIPTION

Makekey improves the usefulness of encryption schemes depending on a key by increasing the amount of time required to search the key space. It reads 10 bytes from its standard input, and writes 13 bytes on its standard output. The output depends on the input in a way intended to be difficult to compute (that is, to require a substantial fraction of a second).

The first eight input bytes (the *input key*) can be arbitrary ASCII characters. The last two (the *salt*) are best chosen from the set of digits, upper- and lower-case letters, and '.' and '/'. The salt characters are repeated as the first two characters of the output. The remaining 11 output characters are chosen from the same set as the salt and constitute the *output key*.

The transformation performed is essentially the following: the salt is used to select one of 4096 cryptographic machines all based on the National Bureau of Standards DES algorithm, but modified in 4096 different ways. Using the input key as key, a constant string is fed into the machine and recirculated a number of times. The 64 bits that come out are distributed into the 66 useful key bits in the result.

Makekey is intended for programs that perform encryption (for instance, *ed* and *crypt*(1)). Usually makekey's input and output will be pipes.

SEE ALSO

crypt(1), ed(1)

STATUS

MAKEKEY(8) currently is not supported by Digital Equipment Corporation.

NAME

mkfs – construct a file system

SYNTAX

/etc/mkfs special size [nsect] [ntrack] [blksize] [fragsize] [ncpg] [minfree] [rps] [nbpi]

DESCRIPTION

File system are normally created with the *newfs*(8) command.

Mkfs constructs a file system by writing on the special file *special.* The numeric size specifies the number of sectors in the file system. *Mkfs* builds a file system with a root directory and a *lost+found* directory. (see *fsck*(8)) The number of i-nodes is calculated as a function of the file system size. No boot program is initialized by *mkfs* (see *newfs*(8).)

The optional arguments allow fine tune control over the parameters of the file system. **Nsect** specify the number of sectors per track on the disk. **Ntrack** specify the number of tracks per cylinder on the disk. **Blksize** gives the primary block size for files on the file system. It must be a power of two, currently selected from 4096 or 8192. **Fragsize** gives the fragment size for files on the file system. The **fragsize** represents the smallest amount of disk space that will be allocated to a file. It must be a power of two currently selected from the range 512 to 8192. **Ncpg** specifies the number of disk cylinders per cylinder group. This number must be in the range 1 to 32. **Minfree** specifies the minimum percentage of free disk space allowed. Once the file system capacity reaches this threshold, only the super-user is allowed to allocate disk blocks. The default value is 10%. If a disk does not revolve at 60 revolutions per second, the **rps** parameter may be specified. Users with special demands for their file systems are referred to the paper cited below for a discussion of the tradeoffs in using different configurations. **nbpi** specifies the number (ratio) of bytes per inode. The default is 2048 bytes.

SEE ALSO

fs(5), dir(5), fsck(8), newfs(8), tunefs(8)

McKusick, Joy, Leffler; "A Fast File System for Unix", Computer Systems Research Group, Dept of EECS, Berkeley, CA 94720; TR #7, September 1982.

STATUS

MKFS(8) is supported by Digital Equipment Corporation.

NAME

mklost+found – make a lost+found directory for fsck

SYNTAX

/etc/mklost+found

DESCRIPTION

A directory *lost+found* is created in the current directory and a number of empty files are created therein and then removed so that there will be empty slots for *fsck*(8). This command should not normally be needed since *mkfs*(8) automatically creates the *lost+found* directory when a new file system is created.

SEE ALSO

fsck(8), mkfs(8)

STATUS

MKLOST+FOUND(8) is supported by Digital Equipment Corporation.

NAME

mknod – build special file

SYNTAX

/etc/mknod name [**c**] [**b**] major minor

DESCRIPTION

Mknod makes a special file. The first argument is the *name* of the entry. The second is **b** if the special file is block-type (disks, tape) or **c** if it is character-type (other devices). The last two arguments are numbers specifying the *major* device type and the *minor* device (e.g. unit, drive, or line number).

The assignment of major device numbers is specific to each system. They have to be dug out of the system source file *conf.c*.

SEE ALSO

mknod(2)

STATUS

MKNOD(8) is supported by Digital Equipment Corporation.

NAME

mkproto – construct a prototype file system

SYNTAX

/etc/mkproto special proto

DESCRIPTION

Mkproto is used to bootstrap a new file system. First a new file system is created using *newfs*(8). *Mkproto* is then used to copy files from the old file system into the new file system according to the directions found in the prototype file *proto*. The prototype file contains tokens separated by spaces or new lines. The first tokens comprise the specification for the root directory. File specifications consist of tokens giving the mode, the user-id, the group id, and the initial contents of the file. The syntax of the contents field depends on the mode.

The mode token for a file is a 6 character string. The first character specifies the type of the file. (The characters **–bcd** specify regular, block special, character special and directory files respectively.) The second character of the type is either **u** or – to specify set-user-id mode or not. The third is **g** or – for the set-group-id mode. The rest of the mode is a three digit octal number giving the owner, group, and other read, write, execute permissions, see *chmod*(1).

Two decimal number tokens come after the mode; they specify the user and group ID's of the owner of the file.

If the file is a regular file, the next token is a pathname whence the contents and size are copied.

If the file is a block or character special file, two decimal number tokens follow which give the major and minor device numbers.

If the file is a directory, *mkproto* makes the entries . and .. and then reads a list of names and (recursively) file specifications for the entries in the directory. The scan is terminated with the token **$**.

A sample prototype specification follows:

```
d--777 3 1
usr     d--777 3 1
        sh      ---755 3 1 /bin/sh
        ken     d--755 6 1
                $
        b0      b--644 3 1 0 0
        c0      c--644 3 1 0 0
        $
$
```

SEE ALSO

fs(5), dir(5), fsck(8), newfs(8)

RESTRICTIONS

Mkproto can only be run on virgin file systems. It should be possible to copy files into existent file systems.

STATUS

MKPROTO(8) currently is not supported by Digital Equipment Corporation.

NAME

mount, umount – mount and dismount file system

SYNTAX

/etc/mount [special name [**–r**]]

/etc/mount –a

/etc/umount special

/etc/umount –a

DESCRIPTION

Mount announces to the system that a removable file system is present on the device *special.* The file *name* must exist already; it must be a directory (unless the root of the mounted file system is not a directory). It becomes the name of the newly mounted root. The optional argument **–r** indicates that the file system is to be mounted read-only.

Umount announces to the system that the removable file system previously mounted on device *special* is to be removed.

If the **–a** option is present for either *mount* or *umount,* all of the file systems described in */etc/fstab* are attempted to be mounted or unmounted. In this case, *special* and *name* are taken from */etc/fstab.* The *special* file name from */etc/fstab* is the block special name.

These commands maintain a table of mounted devices in */etc/mtab.* If invoked without an argument, *mount* prints the table.

Physically write-protected and magnetic tape file systems must be mounted read-only or errors will occur when access times are updated, whether or not any explicit write is attempted.

FILES

/etc/mtab	mount table
/etc/fstab	file system table

SEE ALSO

mount(2), mtab(5), fstab(5)

RESTRICTIONS

Mounting file systems full of garbage will crash the system.

Mounting a root directory on a non-directory makes some apparently good pathnames invalid.

STATUS

MOUNT(8) is supported by Digital Equipment Corporation.

NAME

ncheck – generate names from i-numbers

SYNTAX

/etc/ncheck [**–i** numbers] [**–a**] [**–s**] [filesystem]

DESCRIPTION

For most normal file system maintenance, the function of *ncheck* is subsumed by *fsck*(8).

Ncheck with no argument generates a pathname vs. i-number list of all files on a set of default file systems. Names of directory files are followed by '/.'. The **–i** option reduces the report to only those files whose i-numbers follow. The **–a** option allows printing of the names '.' and '..', which are ordinarily suppressed. The **–s** option reduces the report to special files and files with set-user-ID mode; it is intended to discover concealed violations of security policy.

A file system may be specified.

The report is in no useful order, and probably should be sorted.

SEE ALSO

sort(1), dcheck(8), fsck(8), icheck(8)

DIAGNOSTICS

When the filesystem structure is improper, '??' denotes the 'parent' of a parentless file and a pathname beginning with '...' denotes a loop.

STATUS

NCHECK(8) is supported by Digital Equipment Corporation.

NAME

newfs – construct a new file system

SYNTAX

/etc/newfs [–v] [–n] [mkfs-options] special disk-type

DESCRIPTION

Newfs is a "friendly" front-end to the *mkfs*(8) program. *Newfs* will look up the type of disk a file system is being created on in the disk description file */etc/disktab*, calculate the appropriate parameters to use in calling *mkfs*, then build the file system by forking *mkfs* and, if the file system is a root partition, install the necessary bootstrap programs in the initial 8 sectors of the device. The **–n** option prevents the bootstrap programs from being installed.

If the **–v** option is supplied, *newfs* will print out its actions, including the parameters passed to *mkfs*.

Options which may be used to override default parameters passed to *mkfs* are:

–s size The size of the file system in sectors.

–b block-size
The block size of the file system in bytes.

–f frag-size
The fragment size of the file system in bytes.

–t #tracks/cylinder

–c #cylinders/group
The number of cylinders per cylinder group in a file system. The default value used is 16.

–m free space %
The percentage of space reserved from normal users; the minimum free space threshhold. The default value used is 10%.

–r revolutions/minute
The speed of the disk in revolutions per minute (normally 3600).

–S sector-size
The size of a sector in bytes (almost never anything but 512).

–i number of bytes per inode
This specifies the density of inodes in the file system. The default is to create an inode for each 2048 bytes of data space. If fewer inodes are desired, a larger number should be used; to create more inodes a smaller number should be given.

FILES

/etc/disktab	for disk geometry and file system partition information
/etc/mkfs	to actually build the file system
/usr/mdec	for boot strapping programs

SEE ALSO

disktab(5), fs(5), diskpart(8), fsck(8), format(8), mkfs(8), tunefs(8)

McKusick, Joy, Leffler; "A Fast File System for Unix", Computer Systems Research Group, Dept of EECS, Berkeley, CA 94720; TR #7, September 1982.

STATUS

NEWFS(8) is supported by Digital Equipment Corporation.

NAME

pac – printer/ploter accounting information

SYNTAX

/etc/pac [**–Pprinter**] [**–pprice**] [**–s**] [**–r**] [**–c**] [name ...]

DESCRIPTION

Pac reads the printer/plotter accounting files, accumulating the number of pages (the usual case) or feet (for raster devices) of paper consumed by each user, and printing out how much each user consumed in pages or feet and dollars. If any *names* are specified, then statistics are only printed for those users; usually, statistics are printed for every user who has used any paper.

The **–P** flag causes accounting to be done for the named printer. Normally, accounting is done for the default printer (site dependent) or the value of the environment variable **PRINTER** is used.

The **–p** flag causes the value *price* to be used for the cost in dollars instead of the default value of 0.02.

The **–c** flag causes the output to be sorted by cost; usually the output is sorted alphabetically by name.

The **–r** flag reverses the sorting order.

The **–s** flag causes the accounting information to be summarized on the summary accounting file; this summarization is necessary since on a busy system, the accounting file can grow by several lines per day.

FILES

/usr/adm/?acct	raw accounting files
/usr/adm/?_sum	summary accounting files

RESTRICTIONS

The relationship between the computed price and reality is as yet unknown.

STATUS

PAC(8) currently is not supported by Digital Equipment Corporation.

NAME

pstat – print system facts

SYNTAX

/etc/pstat –aixptufT [suboptions] [system] [corefile]

DESCRIPTION

Pstat interprets the contents of certain system tables. If *corefile* is given, the tables are sought there, otherwise in */dev/kmem*. The required namelist is taken from */vmunix* unless *system* is specified. Options are

–a Under **–p**, describe all process slots rather than just active ones.

–i Print the inode table with the these headings:

LOC The core location of this table entry.

FLAGS Miscellaneous state variables encoded thus:

- **L** locked
- **U** update time (*fs*(5)) must be corrected
- **A** access time must be corrected
- **M** file system is mounted here
- **W** wanted by another process (L flag is on)
- **T** contains a text file
- **C** changed time must be corrected
- **S** shared lock applied
- **E** exclusive lock applied
- **Z** someone waiting for an exclusive lock

CNT Number of open file table entries for this inode.

DEV Major and minor device number of file system in which this inode resides.

RDC Reference count of shared locks on the inode.

WRC Reference count of exclusive locks on the inode (this may be > 1 if, for example, a file descriptor is inherited across a fork).

INO I-number within the device.

MODE Mode bits, see *chmod*(2).

NLK Number of links to this inode.

UID User ID of owner.

SIZ/DEV

Number of bytes in an ordinary file, or major and minor device of special file.

–x Print the text table with these headings:

LOC The core location of this table entry.

FLAGS Miscellaneous state variables encoded thus:

- **T** *ptrace*(2) in effect
- **W** text not yet written on swap device
- **L** loading in progress
- **K** locked
- **w** wanted (L flag is on)
- **P** resulted from demand-page-from-inode exec format (see *execve*(2))

DADDR
Disk address in swap, measured in multiples of 512 bytes.

CADDR
Head of a linked list of loaded processes using this text segment.

SIZE Size of text segment, measured in multiples of 512 bytes.

IPTR Core location of corresponding inode.

CNT Number of processes using this text segment.

CCNT Number of processes in core using this text segment.

-p Print process table for active processes with these headings:

LOC The core location of this table entry.

S Run state encoded thus:
- **0** no process
- **1** waiting for some event
- **3** runnable
- **4** being created
- **5** being terminated
- **6** stopped under trace

F Miscellaneous state variables, or-ed together (hexadecimal):
- **000001** loaded
- **000002** the scheduler process
- **000004** locked for swap out
- **000008** swapped out
- **000010** traced
- **000020** used in tracing
- **000080** in page-wait
- **000100** prevented from swapping during *fork*(2)
- **000200** gathering pages for raw i/o
- **000400** exiting
- **001000** process resulted from a *vfork*(2) which is not yet complete
- **002000** another flag for *vfork*(2)
- **004000** process has no virtual memory, as it is a parent in the context of *vfork*(2)
- **008000** process is demand paging data pages from its text inode.
- **010000** process has advised of anomalous behavior with *vadvise*(2).
- **020000** process has advised of sequential behavior with *vadvise*(2).
- **040000** process is in a sleep which will timeout.
- **080000** a parent of this process has exited and this process is now considered detached.
- **100000** process used 4.1BSD compatibility mode signal primitives, no system calls will restart.
- **200000** process is owed a profiling tick.

POIP number of pages currently being pushed out from this process.

PRI Scheduling priority, see *setpriority*(2).

SIGNAL
Signals received (signals 1-32 coded in bits 0-31),

UID Real user ID.

SLP Amount of time process has been blocked.

TIM Time resident in seconds; times over 127 coded as 127.

CPU Weighted integral of CPU time, for scheduler.

NI Nice level, see *setpriority*(2).

PGRP Process number of root of process group (the opener of the controlling terminal).

PID The process ID number.

PPID The process ID of parent process.

ADDR If in core, the page frame number of the first page of the 'u-area' of the process. If swapped out, the position in the swap area measured in multiples of 512 bytes.

RSS Resident set size – the number of physical page frames allocated to this process.

SRSS RSS at last swap (0 if never swapped).

SIZE Virtual size of process image (data+stack) in multiples of 512 bytes.

WCHAN
Wait channel number of a waiting process.

LINK Link pointer in list of runnable processes.

TEXTP If text is pure, pointer to location of text table entry.

CLKT Countdown for real interval timer, *setitimer*(2) measured in clock ticks (10 milliseconds).

–t Print table for terminals with these headings:

RAW Number of characters in raw input queue.

CAN Number of characters in canonicalized input queue.

OUT Number of characters in putput queue.

MODE See *tty*(4).

ADDR Physical device address.

DEL Number of delimiters (newlines) in canonicalized input queue.

COL Calculated column position of terminal.

STATE Miscellaneous state variables encoded thus:

- **W** waiting for open to complete
- **O** open
- **S** has special (output) start routine
- **C** carrier is on
- **B** busy doing output
- **A** process is awaiting output
- **X** open for exclusive use
- **H** hangup on close

PGRP Process group for which this is controlling terminal.

DISC Line discipline; blank is old tty OTTYDISC or "new tty" for NTTYDISC or "net" for NETLDISC (see *bk*(4)).

–u print information about a user process; the next argument is its address as given

by *ps*(1). The process must be in main memory, or the file used can be a core image and the address 0.

−f Print the open file table with these headings:

LOC The core location of this table entry.

TYPE The type of object the file table entry points to.

FLG Miscellaneous state variables encoded thus:

R open for reading

W open for writing

A open for appending

CNT Number of processes that know this open file.

INO The location of the inode table entry for this file.

OFFS/SOCK

The file offset (see *lseek*(2)), or the core address of the associated socket structure.

−s print information about swap space usage: the number of (1k byte) pages used and free is given as well as the number of used pages which belong to text images.

−T prints the number of used and free slots in the several system tables and is useful for checking to see how full system tables have become if the system is under heavy load.

FILES

/vmunix namelist

/dev/kmem default source of tables

SEE ALSO

ps(1), stat(2), fs(5)

K. Thompson, *UNIX Implementation*

STATUS

PSTAT(8) currently is not supported by Digital Equipment Corporation.

NAME

quot – summarize file system ownership

SYNTAX

/etc/quot [option] ... [filesystem]

DESCRIPTION

Quot prints the number of blocks in the named *filesystem* currently owned by each user. If no *filesystem* is named, a default name is assumed. The following options are available:

–n Cause the pipeline **ncheck filesystem | sort +0n | quot –n filesystem** to produce a list of all files and their owners.

–c Print three columns giving file size in blocks, number of files of that size, and cumulative total of blocks in that size or smaller file.

–f Print count of number of files as well as space owned by each user.

FILES

Default file system varies with system.
/etc/passwd to get user names

SEE ALSO

ls(1), du(1)

STATUS

QUOT(8) currently is not supported by Digital Equipment Corporation.

NAME

quotacheck – file system quota consistency checker

SYNTAX

/etc/quotacheck [**–v**] filesystem...
/etc/quotacheck [**–v**] **–a**

DESCRIPTION

Quotacheck examines each file system, builds a table of current disc usage, and compares this table against that stored in the disc quota file for the file system. If any inconsistencies are detected, both the quota file and the current system copy of the incorrect quotas are updated (the latter only occurs if an active file system is checked).

If the **–a** flag is supplied in place of any file system names, *quotacheck* will check all the file systems indicated in */etc/fstab* to be read-write with disc quotas.

Normally *quotacheck* reports only those quotas modified. If the **–v** option is supplied, *quotacheck* will indicate the calculated disc quotas for each user on a particular file system.

Quotacheck expects each file system to be checked to have a quota file named *quotas* in the root directory. If none is present, *quotacheck* will ignore the file system.

Quotacheck is normally run at boot time from the */etc/rc.local* file, see *rc*(8), before enabling disc quotas with *quotaon*(8).

Quotacheck accesses the raw device in calculating the actual disc usage for each user. Thus, the file systems checked should be quiescent while *quotacheck* is running.

FILES

/etc/fstab default file systems

SEE ALSO

quota(2), setquota(2), quotaon(8)

STATUS

QUOTACHECK(8) currently is not supported by Digital Equipment Corporation.

NAME

quotaon, quotaoff – turn file system quotas on and off

SYNTAX

/etc/quotaon [**–v**] *filsys...*

/etc/quotaon [**–v**] **–a**

/etc/quotaoff [**–v**] *filsys...*

/etc/quotaoff [**–v**] **–a**

DESCRIPTION

Quotaon announces to the system that disc quotas should be enabled on one or more file systems. The file systems specified must have entries in /etc/fstab and be mounted at the time. The file system quota files must be present in the root directory of the specified file system and be named *quotas*. The optional argument **–v** causes *quotaon* to print a message for each file system where quotas are turned on. If, instead of a list of file systems, a **–a** argument is give to *quotaon*, all file systems in /etc/fstab marked read-write with quotas will have their quotas turned on. This is normally used at boot time to enable quotas.

Quotaoff announces to the system that file systems specified should have any disc quotas turned off. As above, the **–v** forces a verbose message for each file system affected; and the **–a** option forces all file systems in /etc/fstab to have their quotas disabled.

These commands update the status field of devices located in */etc/mtab* to indicate when quotas are on or off for each file system.

FILES

/etc/mtab	mount table
/etc/fstab	file system table

SEE ALSO

setquota(2), mtab(5), fstab(5)

STATUS

QUOTAON(8) currently is not supported by Digital Equipment Corporation.

NAME

rc – command script for auto-reboot and daemons

SYNTAX

/etc/rc
/etc/rc.local

DESCRIPTION

Rc is the command script which controls the automatic reboot and *rc.local* is the script holding commands which are pertinent only to a specific site.

When an automatic reboot is in progress, *rc* is invoked with the argument *autoboot* and runs a *fsck* with option **–p** to "preen" all the disks of minor inconsistencies resulting from the last system shutdown and to check for serious inconsistencies caused by hardware or software failure. If this auto-check and repair succeeds, then the second part of *rc* is run.

The second part of *rc,* which is run after a auto-reboot succeeds and also if *rc* is invoked when a single user shell terminates (see *init*(8)), starts all the daemons on the system, preserves editor files and clears the scratch directory **/tmp.** *Rc.local* is executed immediately before any other commands after a successful *fsck*. Normally, the first commands placed in the *rc.local* file define the machine's name, using *hostname*(1), and save any possible core image that might have been generated as a result of a system crash, *savecore*(8). The latter command is included in the *rc.local* file because the directory in which core dumps are saved is usually site specific.

SEE ALSO

init(8), reboot(8), savecore(8)

STATUS

RC(8) currently is not supported by Digital Equipment Corporation.

NAME

rdump – file system dump across the network

SYNTAX

/etc/rdump [key [*argument* ...] filesystem]

DESCRIPTION

Rdump copies to magnetic tape all files changed after a certain date in the *filesystem.* The command is identical in operation to *dump*(8) except the *f* key should be specified and the file supplied should be of the form *machine:device.*

Rdump creates a remote server, */etc/rmt*, on the client machine to access the tape device.

SEE ALSO

dump(8), rmt(8C)

DIAGNOSTICS

Same as *dump*(8) with a few extra related to the network.

STATUS

RDUMP(8C) currently is not supported by Digital Equipment Corporation.

NAME

reboot – UNIX bootstrapping procedures

SYNTAX

/etc/reboot [**–n**] [**–q**]

DESCRIPTION

UNIX is started by placing it in memory at location zero and transferring to zero. Since the system is not reenterable, it is necessary to read it in from disk or tape each time it is to be bootstrapped.

Rebooting a running system. When a UNIX is running and a reboot is desired, *shutdown*(8) is normally used. If there are no users then **/etc/reboot** can be used. Reboot causes the disks to be synced, and then a multi-user reboot (as described below) is initiated. This causes a system to be booted and an automatic disk check to be performed. If all this succeeds without incident, the system is then brought up for many users.

Options to reboot are:

–n option avoids the sync. It can be used if a disk or the processor is on fire.

–q reboots quickly and ungracefully, without shutting down running processes first.

Power fail and crash recovery. Normally, the system will reboot itself at power-up or after crashes. Provided the auto-restart is enabled on the machine front panel, an automatic consistency check of the file systems will be performed then and unless this fails the system will resume multi-user operations.

Cold starts. These are processor type dependent. On an 11/780, there are two floppy files for each disk controller, both of which cause boots from unit 0 of the root file system of a controller located on mba0 or uba0. One gives a single user shell, while the other invokes the multi-user automatic reboot. Thus these files are HPS and HPM for the single and multi-user boot from MASSBUS RP06/RM03/RM05 disks, UPS and UPM for UNIBUS storage module controller and disks such as the EMULEX SC-21 and AMPEX 9300 pair, or HKS and HKM for RK07 disks.

Giving the command

>>>BOOT HPM

Would boot the system from (e.g.) an RP06 and run the automatic consistency check as described in *fsck*(8). (Note that it may be necessary to type control-P to gain the attention of the LSI-11 before getting the >>> prompt.) The command

>>>BOOT ANY

invokes a version of the boot program in a way which allows you to specify any system as the system to be booted. It reads from the console a device specification (see below) followed immediately by a pathname.

On an 11/750, the reset button will boot from the device selected by the front panel boot device switch. In systems with RK07's, position B normally selects the RK07 for boot. This will boot multi-user. To boot from RK07 with boot flags you may specify

>>>B/*n* DMA0

where, giving a *n* of 1 causes the boot program to ask for the name of the system to be bootstrapped, giving a *n* of 2 causes the boot program to come up single user, and a *n* of 3 causes both of these actions to occur.

The 11/750 boot procedure uses the boot roms to load block 0 off of the specified device. The /usr/mdec directory contains a number of bootstrap programs for the various disks which should be placed in a new pack automatically by *newfs*(8) when the "a" partition file system on the pack is created.

On both processors, the *boot* program finds the corresponding file on the given device, loads that file into memory location zero, and starts the program at the entry address specified in the program header (after clearing off the high bit of the specified entry address.) Normal line editing characters can be used in specifying the pathname.

If you have a MASSBUS disk and wish to boot off of a file system which starts at cylinder 0 of unit 0, you can type "hp(0,0)vmunix" to the boot prompt; "up(0,0)vmunix" would specify a UNIBUS drive, "hk(0,0)vmunix" would specify an RK07 disk drive, "ra(0,0)vmunix" would specify a UDA50 disk drive, and "rb(0,0)vmunix" would specify a disk on a 730 IDC.

A device specification has the following form:

device(unit, minor)

where *device* is the type of the device to be searched, *unit* is 8* the mba or uba number plus the unit number of the device, and *minor* is the minor device index. The following list of supported devices may vary from installation to installation:

hp	MASSBUS disk drive
up	UNIBUS storage module drive
ht	TE16,TU45,TU77 on MASSBUS
mt	TU78 on MASSBUS
hk	RK07 on UNIBUS
ra	storage module on a UDA50
rb	storage module on a 730 IDC
rl	RL02 on UNIBUS
tm	TM11 emulation tape drives on UNIBUS
ts	TS11 on UNIBUS
ut	UNIBUS TU45 emulator

For tapes, the minor device number gives a file offset.

In an emergency, the bootstrap methods described in the paper "Installing and Operating 4.2bsd" can be used to boot from a distribution tape.

FILES

/vmunix	system code
/boot	system bootstrap

SEE ALSO

crash(8V), fsck(8), init(8), rc(8), shutdown(8), halt(8), newfs(8)

STATUS

REBOOT(8) currently is not supported by Digital Equipment Corporation.

NAME

renice – alter priority of running processes

SYNTAX

/etc/renice priority [[**–p**] pid ...] [[**–g**] pgrp ...] [[**–u**] user ...]

DESCRIPTION

Renice alters the scheduling priority of one or more running processes. The *who* parameters are interpreted as process ID's, process group ID's, or user names. *Renice*'ing a process group causes all processes in the process group to have their scheduling priority altered. *Renice*'ing a user causes all processes owned by the user to have their scheduling priority altered. By default, the processes to be affected are specified by their process ID's. To force *who* parameters to be interpreted as process group ID's, a **–g** may be specified. To force the *who* parameters to be interpreted as user names, a **–u** may be given. Supplying **–p** will reset *who* interpretation to be (the default) process ID's. For example,

```
/etc/renice +1 987 -u daemon root -p 32
```

would change the priority of process ID's 987 and 32, and all processes owned by users daemon and root.

Users other than the super-user may only alter the priority of processes they own, and can only monotonically increase their "nice value" within the range 0 to PRIO_MIN (20). (This prevents overriding administrative fiats.) The super-user may alter the priority of any process and set the priority to any value in the range PRIO_MAX (–20) to PRIO_MIN. Useful priorities are: 19 (the affected processes will run only when nothing else in the system wants to), 0 (the "base" scheduling priority), anything negative (to make things go very fast).

FILES

/etc/passwd to map user names to user ID's

SEE ALSO

getpriority(2), setpriority(2)

RESTRICTIONS

If you make the priority very negative, then the process cannot be interrupted. To regain control you make the priority greater than zero. Non super-users can not increase scheduling priorities of their own processes, even if they were the ones that decreased the priorities in the first place.

STATUS

RENICE(8) currently is not supported by Digital Equipment Corporation.

NAME

repquota – summarize quotas for a file system

SYNTAX

repquota *filesys...*

DESCRIPTION

Repquota prints a summary of the disc usage and quotas for the specified file systems. For each user the current number files and amount of space (in kilobytes) is printed, along with any quotas created with *edquota*(8).

Only the super-user may view quotas which are not their own.

FILES

quotas at the root of each file system with quotas
/etc/fstab for file system names and locations

SEE ALSO

quota(1), quota(2), quotacheck(8), quotaon(8), edquota(8)

DIAGNOSTICS

Various messages about inaccessible files; self-explanatory.

STATUS

REPQUOTA(8) currently is not supported by Digital Equipment Corporation.

NAME

restore – incremental file system restore

SYNTAX

/etc/restore key [name ...]

DESCRIPTION

Restore reads tapes dumped with the *dump*(8) command. Its actions are controlled by the *key* argument. The *key* is a string of characters containing at most one function letter and possibly one or more function modifiers. Other arguments to the command are file or directory names specifying the files that are to be restored. Unless the **h** key is specified (see below), the appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

The function portion of the key is specified by one of the following letters:

r The tape is read and loaded into the current directory. This should not be done lightly; the **r** key should only be used to restore a complete dump tape onto a clear file system or to restore an incremental dump tape after a full level zero restore. Thus

```
/etc/newfs /dev/rrp0g eagle
/etc/mount /dev/rp0g /mnt
cd /mnt
restore r
```

is a typical sequence to restore a complete dump. Another *restore* can be done to get an incremental dump in on top of this. Note that *restore* leaves a file *restoresymtab* in the root directory to pass information between incremental restore passes. This file should be removed when the last incremental tape has been restored.
A *dump*(8) followed by a *newfs*(8) and a *restore* is used to change the size of a file system.

R *Restore* requests a particular tape of a multi volume set on which to restart a full restore (see the **r** key above). This allows *restore* to be interrupted and then restarted.

x The named files are extracted from the tape. If the named file matches a directory whose contents had been written onto the tape, and the **h** key is not specified, the directory is recursively extracted. The owner, modification time, and mode are restored (if possible). If no file argument is given, then the root directory is extracted, which results in the entire content of the tape being extracted, unless the **h** key has been specified.

t The names of the specified files are listed if they occur on the tape. If no file argument is given, then the root directory is listed, which results in the entire content of the tape being listed, unless the **h** key has been specified. Note that the **t** key replaces the function of the old *dumpdir* program.

i This mode allows interactive restoration of files from a dump tape. After reading in

the directory information from the tape, *restore* provides a shell like interface that allows the user to move around the directory tree selecting files to be extracted. The available commands are given below; for those commands that require an argument, the default is the current directory.

ls [arg] – List the current or specified directory. Entries that are directories are appended with a "/". Entries that have been marked for extraction are prepended with a "*". If the verbose key is set the inode number of each entry is also listed.

cd arg – Change the current working directory to the specified argument.

pwd – Print the full pathname of the current working directory.

add [arg] – The current directory or specified argument is added to the list of files to be extracted. If a directory is specified, then it and all its descendents are added to the extraction list (unless the **h** key is specified on the command line). Files that are on the extraction list are prepended with a "*" when they are listed by **ls**.

delete [arg] – The current directory or specified argument is deleted from the list of files to be extracted. If a directory is specified, then it and all its descendents are deleted from the extraction list (unless the **h** key is specified on the command line). The most expedient way to extract most of the files from a directory is to add the directory to the extraction list and then delete those files that are not needed.

extract – All the files that are on the extraction list are extracted from the dump tape. *Restore* will ask which volume the user wishes to mount. The fastest way to extract a few files is to start with the last volume, and work towards the first volume.

verbose – The sense of the **v** key is toggled. When set, the verbose key causes the **ls** command to list the inode numbers of all entries. It also causes *restore* to print out information about each file as it is extracted.

help – List a summary of the available commands.

quit – Restore immediately exits, even if the extraction list is not empty.

The following characters may be used in addition to the letter that selects the function desired.

v Normally *restore* does its work silently. The **v** (verbose) key causes it to type the

name of each file it treats preceded by its file type.

f The next argument to *restore* is used as the name of the archive instead of /dev/rmt?. If the name of the file is "−", *restore* reads from standard input. Thus, *dump*(8) and *restore* can be used in a pipeline to dump and restore a file system with the command

dump 0f - /usr | (cd /mnt; restore xf -)

y *Restore* will not ask whether it should abort the restore if gets a tape error. It will always try to skip over the bad tape block(s) and continue as best it can.

m *Restore* will extract by inode numbers rather than by file name. This is useful if only a few files are being extracted, and one wants to avoid regenerating the complete pathname to the file.

h *Restore* extracts the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the tape.

DIAGNOSTICS

Complaints about bad key characters.

Complaints if it gets a read error. If **y** has been specified, or the user responds "y", *restore* will attempt to continue the restore.

If the dump extends over more than one tape, *restore* will ask the user to change tapes. If the **x** or **i** key has been specified, *restore* will also ask which volume the user wishes to mount. The fastest way to extract a few files is to start with the last volume, and work towards the first volume.

There are numerous consistency checks that can be listed by *restore*. Most checks are self-explanatory or can "never happen". Common errors are given below.

Converting to new file system format.
A dump tape created from the old file system has been loaded. It is automatically converted to the new file system format.

<filename>: not found on tape
The specified file name was listed in the tape directory, but was not found on the tape. This is caused by tape read errors while looking for the file, and from using a dump tape created on an active file system.

expected next file <inumber>, got <inumber>
A file that was not listed in the directory showed up. This can occur when using a dump tape created on an active file system.

Incremental tape too low
When doing incremental restore, a tape that was written before the previous incremental tape, or that has too low an incremental level has been loaded.

Incremental tape too high
When doing incremental restore, a tape that does not begin its coverage where the previous incremental tape left off, or that has too high an incremental level has been loaded.

Tape read error while restoring <filename>
Tape read error while skipping over inode <inumber>
Tape read error while trying to resynchronize

A tape read error has occurred. If a file name is specified, then its contents are probably partially wrong. If an inode is being skipped or the tape is trying to resynchronize, then no extracted files have been corrupted, though files may not be found on the tape.

resync restore, skipped <num> blocks

After a tape read error, *restore* may have to resynchronize itself. This message lists the number of blocks that were skipped over.

FILES

/dev/rmt?	the default tape drive
/tmp/rstdir*	file containing directories on the tape.
/tmp/rstmode*	owner, mode, and time stamps for directories.
./restoresymtab	information passed between incremental restores.

SEE ALSO

rrestore(8C) dump(8), newfs(8), mount(8), mkfs(8)

RESTRICTIONS

Restore can get confused when doing incremental restores from dump tapes that were made on active file systems.

A level zero dump must be done after a full restore. Because restore runs in user code, it has no control over inode allocation; thus a full restore must be done to get a new set of directories reflecting the new inode numbering, even though the contents of the files is unchanged.

STATUS

RESTORE(8) is supported by Digital Equipment Corporation.

NAME

rexecd – remote execution server

SYNTAX

/etc/rexecd

DESCRIPTION

Rexecd is the server for the *rexec*(3X) routine. The server provides remote execution facilities with authentication based on user names and encrypted passwords.

Rexecd listens for service requests at the port indicated in the "exec" service specification; see *services*(5). When a service request is received the following protocol is initiated:

1) The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.

2) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine.

3) A null terminated user name of at most 16 characters is retrieved on the initial socket.

4) A null terminated, encrypted, password of at most 16 characters is retrieved on the initial socket.

5) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

6) *Rexecd* then validates the user as is done at login time and, if the authentication was successful, changes to the user's home directory, and establishes the user and group protections of the user. If any of these steps fail the connection is aborted with a diagnostic message returned.

7) A null byte is returned on the connection associated with the **stderr** and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by *rexecd*.

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

"username too long"
The name is longer than 16 characters.

"password too long"
The password is longer than 16 characters.

"command too long "
The command line passed exceeds the size of the argument list (as configured into the system).

"Login incorrect."
No password file entry for the user name existed.

"Password incorrect."
The wrong was password supplied.

"No remote directory."
The *chdir* command to the home directory failed.

"Try again."
A *fork* by the server failed.

"/bin/sh: ..."
The user's login shell could not be started.

RESTRICTIONS

Indicating "Login incorrect" as opposed to "Password incorrect" is a security breach which allows people to probe a system for users with null passwords.

STATUS

REXECD(8C) currently is not supported by Digital Equipment Corporation.

NAME

rlogind – remote login server

SYNTAX

/etc/rlogind [**–d**]

DESCRIPTION

Rlogind is the server for the *rlogin*(1C) program. The server provides a remote login facility with authentication based on privileged port numbers.

Rlogind listens for service requests at the port indicated in the "login" service specification; see *services*(5). When a service request is received the following protocol is initiated:

1) The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection.

2) The server checks the client's source address. If the address is associated with a host for which no corresponding entry exists in the host name data base (see *hosts*(5)), the server aborts the connection.

Once the source port and address have been checked, *rlogind* allocates a pseudo terminal (see *pty*(4)), and manipulates file descriptors so that the slave half of the pseudo terminal becomes the **stdin , stdout ,** and **stderr** for a login process. The login process is an instance of the *login*(1) program, invoked with the **–r** option. The login process then proceeds with the authentication process as described in *rshd*(8C), but if automatic authentication fails, it reprompts the user to login as one finds on a standard terminal line.

The parent of the login process manipulates the master side of the pseduo terminal, operating as an intermediary between the login process and the client instance of the *rlogin* program. In normal operation, the packet protocol described in *pty*(4) is invoked to provide ^S/^Q type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the environment variable, "TERM"; see *environ*(7).

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1.

"Hostname for your address unknown."
No entry in the host name database existed for the client's machine.

"Try again."
A *fork* by the server failed.

"/bin/sh: ..."
The user's login shell could not be started.

RESTRICTIONS

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.

STATUS

RLOGIND(8C) is supported by Digital Equipment Corporation.

NAME

rmt – remote magtape protocol module

SYNTAX

/etc/rmt

DESCRIPTION

Rmt is a program used by the remote dump and restore programs in manipulating a magnetic tape drive through an interprocess communication connection. *Rmt* is normally started up with an *rexec*(3X) or *rcmd*(3X) call.

The *rmt* program accepts requests specific to the manipulation of magnetic tapes, performs the commands, then responds with a status indication. All responses are in ASCII and in one of two forms. Successful commands have responses of

A*number*\n

where *number* is an ASCII representation of a decimal number. Unsuccessful commands are responded to with

E*error-number*\n*error-message*\n,

where *error-number* is one of the possible error numbers described in *intro*(2) and *error-message* is the corresponding error string as printed from a call to *perror*(3). The protocol is comprised of the following commands (a space is present between each token).

O device mode
: Open the specified *device* using the indicated *mode*. *Device* is a full pathname and *mode* is an ASCII representation of a decimal number suitable for passing to *open*(2). If a device had already been opened, it is closed before a new open is performed.

C device
: Close the currently open device. The *device* specified is ignored.

L whence offset
: Perform an *lseek*(2) operation using the specified parameters. The response value is that returned from the *lseek* call.

W count
: Write data onto the open device. *Rmt* reads *count* bytes from the connection, aborting if a premature end-of-file is encountered. The response value is that returned from the *write*(2) call.

R count
: Read *count* bytes of data from the open device. If *count* exceeds the size of the data buffer (10 kilobytes), it is truncated to the data buffer size. *Rmt* then performs the requested *read*(2) and responds with **A***count-read*\n if the read was successful; otherwise an error in the standard format is returned. If the read was successful, the data read is then sent.

I operation count
: Perform a MTIOCOP *ioctl*(2) command using the specified parameters. The parameters are interpreted as the ASCII representations of the decimal values to place in the *mt_op* and *mt_count* fields of the structure

used in the *ioctl* call. The return value is the *count* parameter when the operation is successful.

S Return the status of the open device, as obtained with a MTIOCGET *ioctl* call. If the operation was successful, an "ack" is sent with the size of the status buffer, then the status buffer is sent (in binary).

Any other command causes *rmt* to exit.

DIAGNOSTICS

All responses are of the form described above.

SEE ALSO

rcmd(3X), rexec(3X), mtio(4), rdump(8C), rrestore(8C)

RESTRICTIONS

People tempted to use this for a remote file access protocol are discouraged.

STATUS

RMT(8C) currently is not supported by Digital Equipment Corporation.

NAME

route – manually manipulate the routing tables

SYNTAX

/etc/route [**–f**] [*command args*]

DESCRIPTION

Route is a program used to manually manipulate the network routing tables. It normally is not needed, as the system routing table management daemon, *routed*(8C), should tend to this task.

Route accepts three commands: *add*, to add a route; *delete*, to delete a route; and *change*, to modify an existing route.

All commands have the following syntax:

/etc/route *command* **destination gateway** [**metric**]

where *destination* is a host or network for which the route is "to", *gateway* is the gateway to which packets should be addressed, and *metric* is an optional count indicating the number of hops to the *destination*. If no metric is specified, *route* assumes a value of 0. Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination*. If the *destination* has a "local address part" of INADDR ANY, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host. If the route is to a destination connected via a gateway, the *metric* should be greater than 0. All symbolic names specified for a *destination* or *gateway* are looked up first in the host name database, *hosts*(5). If this lookup fails, the name is then looked for in the network name database, *networks*(5).

Route uses a raw socket and the SIOCADDRT and SIOCDELRT *ioctl*'s to do its work. As such, only the super-user may modify the routing tables.

If the **–f** option is specified, *route* will "flush" the routing tables of all gateway entries. If this is used in conjunction with one of the commands described above, the tables are flushed prior to the command's application.

DIAGNOSTICS

"add %s: gateway %s flags %x"
The specified route is being added to the tables. The values printed are from the routing table entry supplied in the *ioctl* call.

"delete %s: gateway %s flags %x"
As above, but when deleting an entry.

"%s %s done"
When the **–f** flag is specified, each routing table entry deleted is indicated with a message of this form.

"not in table"
A delete operation was attempted for an entry which wasn't present in the tables.

"routing table overflow"
An add operation was attempted, but the system was low on resources and was unable to allocate memory to create the new entry.

SEE ALSO

intro(4N), routed(8C)

RESTRICTIONS

The change operation is not implemented. Therefore, one should add the new route, then one should delete the old one.

STATUS

ROUTE(8C) currently is not supported by Digital Equipment Corporation.

NAME

routed – network routing daemon

SYNTAX

/etc/routed [**–s**] [**–q**] [**–t**] [*logfile*]

DESCRIPTION

Routed is invoked at boot time to manage the network routing tables. The routing daemon uses a variant of the Xerox NS Routing Information Protocol in maintaining up to date kernel routing table entries.

In normal operation *routed* listens on *udp*(4P) socket 520 (decimal) for routing information packets. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When *routed* is started, it uses the SIOCGIFCONF *ioctl* to find those directly connected interfaces configured into the system and marked "up" (the software loopback interface is ignored). If multiple interfaces are present, it is assumed the host will forward packets between networks. *Routed* then transmits a *request* packet on each interface (using a broadcast packet if the interface supports it) and enters a loop, listening for *request* and *response* packets from other hosts.

When a *request* packet is received, *routed* formulates a reply based on the information maintained in its internal tables. The *response* packet generated contains a list of known routes, each marked with a "hop count" metric (a count of 16, or greater, is considered "infinite"). The metric associated with each route returned provides a metric *relative to the sender*.

Response packets received by *routed* are used to update the routing tables if one of the following conditions is satisfied:

(1) No routing table entry exists for the destination network or host, and the metric indicates the destination is "reachable" (i.e. the hop count is not infinite).

(2) The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.

(3) The existing entry in the routing table has not been updated for some time (defined to be 90 seconds) and the route is at least as cost effective as the current route.

(4) The new route describes a shorter route to the destination than the one currently stored in the routing tables; the metric of the new route is compared against the one stored in the table to decide this.

When an update is applied, *routed* records the change in its internal tables and generates a *response* packet to all directly connected hosts and networks. *Routed* waits a short period of time (no more than 30 seconds) before modifying the kernel's routing tables to allow possible unstable situations to settle.

In addition to processing incoming packets, *routed* also periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to insure the invalidation is propagated throughout the internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks.

Supplying the **−s** option forces *routed* to supply routing information whether it is acting as an internetwork router or not. The **−q** option is the opposite of the **−s** option. If the **−t** option is specified, all packets sent or received are printed on the standard output. In addition, *routed* will not divorce itself from the controlling terminal so that interrupts from the keyboard will kill the process. Any other argument supplied is interpreted as the name of file in which *routed*'s actions should be logged. This log contains information about any changes to the routing tables and a history of recent messages sent and received which are related to the changed route.

In addition to the facilities described above, *routed* supports the notion of "distant" *passive* and *active* gateways. When *routed* is started up, it reads the file */etc/gateways* to find gateways which may not be identified using the SIOGIFCONF *ioctl*. Gateways specified in this manner should be marked passive if they are not expected to exchange routing information, while gateways marked active should be willing to exchange routing information (i.e. they should have a *routed* process running on the machine). Passive gateways are maintained in the routing tables forever and information regarding their existence is included in any routing information transmitted. Active gateways are treated equally to network interfaces. Routing information is distributed to the gateway and if no routing information is received for a period of the time, the associated route is deleted.

The */etc/gateways* is comprised of a series of lines, each in the following format:

< **net** | **host** > *name1* **gateway** *name2* **metric** *value* < **passive** | **active** >

The **net** or **host** keyword indicates if the route is to a network or specific host.

Name1 is the name of the destination network or host. This may be a symbolic name located in */etc/networks* or */etc/hosts*, or an Internet address specified in "dot" notation; see *inet*(3N).

Name2 is the name or address of the gateway to which messages should be forwarded.

Value is a metric indicating the hop count to the destination host or network.

The keyword **passive** or **active** indicates if the gateway should be treated as *passive* or *active* (as described above).

FILES

/etc/gateways for distant gateways

SEE ALSO

"Internet Transport Protocols", XSIS 028112, Xerox System Integration Standard.
udp(4P)

RESTRICTIONS

The kernel's routing tables may not correspond to those of *routed* for short periods of time while processes utilizing existing routes exit; the only remedy for this is to place the routing process in the kernel.

Routed should listen to intelligent interfaces, such as an IMP, and to error protocols, such as ICMP, to gather more information.

STATUS

ROUTED(8C) currently is not supported by Digital Equipment Corporation.

NAME

rrestore – restore a file system dump across the network

SYNTAX

/etc/rrestore [key [name ...]

DESCRIPTION

Rrestore obtains from magnetic tape files saved by a previous *dump*(8). The command is identical in operation to *restore*(8) except the *f* key should be specified and the file supplied should be of the form *machine:device*.

Rrestore creates a remote server, */etc/rmt*, on the client machine to access the tape device.

SEE ALSO

restore(8), rmt(8C)

DIAGNOSTICS

Same as *restore*(8) with a few extra related to the network.

STATUS

RRESTORE(8C) currently is not supported by Digital Equipment Corporation.

NAME

rshd – remote shell server

SYNTAX

/etc/rshd

DESCRIPTION

Rshd is the server for the *rcmd*(3X) routine and, consequently, for the *rsh*(1C) program. The server provides remote execution facilities with authentication based on privileged port numbers.

Rshd listens for service requests at the port indicated in the "cmd" service specification; see *services*(5). When a service request is received the following protocol is initiated:

1) The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection.

2) The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.

3) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine. The source port of this second connection is also in the range 0-1023.

4) The server checks the client's source address. If the address is associated with a host for which no corresponding entry exists in the host name data base (see *hosts*(5)), the server aborts the connection.

5) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as a user identity to use on the **server**'s machine.

6) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as the user identity on the **client**'s machine.

7) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

8) *Rshd* then validates the user according to the following steps. The remote user name is looked up in the password file and a *chdir* is performed to the user's home directory. If either the lookup or *chdir* fail, the connection is terminated. If the user is not the super-user, (user id 0), the file */etc/hosts.equiv* is consulted for a list of hosts considered "equivalent". If the client's host name is present in this file, the authentication is considered successful. If the lookup fails, or the user is the super-user, then the file *.rhosts* in the home directory of the remote user is checked for the machine name and identity of the user on the client's machine. If this lookup fails, the connection is terminated.

9) A null byte is returned on the connection associated with the **stderr** and the command line is passed to the normal login shell of the user. The shell inherits the

network connections established by *rshd*.

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 9 above upon successful completion of all the steps prior to the command execution).

"locuser too long"
The name of the user on the client's machine is longer than 16 characters.

"remuser too long"
The name of the user on the remote machine is longer than 16 characters.

"command too long "
The command line passed exceeds the size of the argument list (as configured into the system).

"Hostname for your address unknown."
No entry in the host name database existed for the client's machine.

"Login incorrect."
No password file entry for the user name existed.

"No remote directory."
The *chdir* command to the home directory failed.

"Permission denied."
The authentication procedure described above failed.

"Can't make pipe."
The pipe needed for the **stderr**, wasn't created.

"Try again."
A *fork* by the server failed.

"/bin/sh: ..."
The user's login shell could not be started.

SEE ALSO

rsh(1C), rcmd(3X)

RESTRICTIONS

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.

STATUS

RSHD(8C) is supported by Digital Equipment Corporation.

NAME

rwhod – system status server

SYNTAX

/etc/rwhod

DESCRIPTION

Rwhod is the server which maintains the database used by the *rwho*(1C) and *ruptime*(1C) programs. Its operation is predicated on the ability to *broadcast* messages on a network.

Rwhod operates as both a producer and consumer of status information. As a producer of information it periodically queries the state of the system and constructs status messages which are broadcast on a network. As a consumer of information, it listens for other *rwhod* servers' status messages, validating them, then recording them in a collection of files located in the directory */usr/spool/rwho*.

The *rwho* server transmits and receives messages at the port indicated in the "rwho" service specification, see *services*(5). The messages sent and received, are of the form:

```
struct  outmp {
        char    out_line[8];/* tty name */
        char    out_name[8];/* user id */
        long    out_time;/* time on */
};

struct  whod {
        char    wd_vers;
        char    wd_type;
        char    wd_fill[2];
        int     wd_sendtime;
        int     wd_recvtime;
        char    wd_hostname[32];
        int     wd_loadav[3];
        int     wd_boottime;
        struct  whoent {
                struct  outmp we_utmp;
                int     we_idle;
        } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order prior to transmission. The load averages are as calculated by the *w*(1) program, and represent load averages over the 5, 10, and 15 minute intervals prior to a server's transmission. The host name included is that returned by the *gethostname*(2) system call. The array at the end of the message contains information about the users logged in to the sending machine. This information includes the contents of the *utmp*(5) entry for each non-idle terminal line and a value indicating the time since a character was last received on the terminal line.

Messages received by the *rwho* server are discarded unless they originated at a *rwho* server's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by *rwhod* are placed in files named *whod.hostname* in the directory */usr/spool/rwho*. These files contain only the most recent message, in the format described above.

Status messages are generated approximately once every 60 seconds. *Rwhod* performs an *nlist*(3) on /vmunix every 10 minutes to guard against the possibility that this file is not the system image currently operating.

SEE ALSO

rwho(1C), ruptime(1C)

STATUS

RWHOD(8C) currently is not supported by Digital Equipment Corporation.

NAME

rxformat – format floppy disks

SYNTAX

/etc/rxformat [**–d**] special

DESCRIPTION

The *rxformat* program formats a diskette in the specified drive associated with the special device *special*. (*Special* is normally /dev/rrx0, for drive 0, or /dev/rrx1, for drive 1.) By default, the diskette is formatted single density; a **–d** flag may be supplied to force double density formatting. Single density is compatible with the IBM 3740 standard (128 bytes/sector). In double density, each sector contains 256 bytes of data.

Before formatting a diskette *rxformat* prompts for verification (this allows a user to cleanly abort the operation; note that formatting a diskette will destroy any existing data). Formatting is done by the hardware. All sectors are zero-filled.

DIAGNOSTICS

'No such device' means that the drive is not ready, usually because no disk is in the drive or the drive door is open. Other error messages are selfexplanatory.

FILES

/dev/rx?

SEE ALSO

rx(4V)

RESTRICTIONS

A floppy may not be formatted if the header info on sector 1, track 0 has been damaged. Hence, it is not possible to format a completely degaussed disk. (This is actually a problem in the hardware.)

AUTHOR

Helge Skrivervik

STATUS

RXFORMAT(8V) currently is not supported by Digital Equipment Corporation.

NAME

sa, accton – system accounting

SYNTAX

/etc/sa [**–abcdDfijkKlnrstuv**] [file]

/etc/accton [file]

DESCRIPTION

With an argument naming an existing *file, accton* causes system accounting information for every process executed to be placed at the end of the file. If no argument is given, accounting is turned off.

Sa reports on, cleans up, and generally maintains accounting files.

Sa is able to condense the information in */usr/adm/acct* into a summary file */usr/adm/savacct* which contains a count of the number of times each command was called and the time resources consumed. This condensation is desirable because on a large system */usr/adm/acct* can grow by 100 blocks per day. The summary file is normally read before the accounting file, so the reports include all available information.

If a file name is given as the last argument, that file will be treated as the accounting file; */usr/adm/acct* is the default.

Output fields are labeled: "cpu" for the sum of user+system time (in minutes), "re" for real time (also in minutes), "k" for cpu-time averaged core usage (in 1k units), "avio" for average number of i/o operations per execution. With options fields labeled "tio" for total i/o operations, "k*sec" for cpu storage integral (kilo-core seconds), "u" and "s" for user and system cpu time alone (both in minutes) will sometimes appear.

There are near a googol of options:

a Place all command names containing unprintable characters and those used only once under the name '***other.'

b Sort output by sum of user and system time divided by number of calls. Default sort is by sum of user and system times.

c Besides total user, system, and real time for each command print percentage of total time over all commands.

d Sort by average number of disk i/o operations.

D Print and sort by total number of disk i/o operations.

f Force no interactive threshold compression with –v flag.

i Don't read in summary file.

j Instead of total minutes time for each category, give seconds per call.

k Sort by cpu-time average memory usage.

K Print and sort by cpu-storage integral.

l	Separate system and user time; normally they are combined.
m	Print number of processes and number of CPU minutes for each user.
n	Sort by number of calls.
r	Reverse order of sort.
s	Merge accounting file into summary file */usr/adm/savacct* when done.
t	For each command report ratio of real time to the sum of user and system times.
u	Superseding all other flags, print for each command in the accounting file the user ID and command name.
v	Followed by a number *n*, types the name of each command used *n* times or fewer. Await a reply from the terminal; if it begins with 'y', add the command to the category '**junk**.' This is used to strip out garbage.

FILES

/usr/adm/acct	raw accounting
/usr/adm/savacct	summary
/usr/adm/usracct	per-user summary

SEE ALSO

ac(8), acct(2)

STATUS

SA(8) currently is not supported by Digital Equipment Corporation.

NAME

savecore – save a core dump of the operating system

SYNTAX

/etc/savecore *dirname* [*system*]

DESCRIPTION

Savecore is meant to be called near the end of the /etc/rc file. Its function is to save the core dump of the system (assuming one was made) and to write a reboot message in the shutdown log.

Savecore checks the core dump to be certain it corresponds with the current running unix. If it does it saves the core image in the file *dirname*/vmcore.n and it's brother, the namelist, *dirname*/vmunix.n The trailing ".n" in the pathnames is replaced by a number which grows every time *savecore* is run in that directory.

Before savecore writes out a core image, it reads a number from the file *dirname*/minfree. If there are fewer free blocks on the filesystem which contains *dirname* than the number obtained from the minfree file, the core dump is not done. If the minfree file does not exist, savecore always writes out the core file (assuming that a core dump was taken).

Savecore also writes a reboot message in the shut down log. If the system crashed as a result of a panic, *savecore* records the panic string in the shut down log too.

If the core dump was from a system other than /vmunix, the name of that system must be supplied as *sysname*.

FILES

/usr/adm/shutdownlog shut down log
/vmunix current UNIX

RESTRICTIONS

Can be fooled into thinking a core dump is the wrong size.

STATUS

SAVECORE(8) currently is not supported by Digital Equipment Corporation.

NAME

sendmail – send mail over the internet

SYNTAX

/usr/lib/sendmail [flags] [address ...]

newaliases

mailq

DESCRIPTION

Sendmail sends a message to one or more people, routing the message over whatever networks are necessary. *Sendmail* does internetwork forwarding as necessary to deliver the message to the correct place.

Sendmail is not intended as a user interface routine; other programs provide user-friendly front ends; *sendmail* is used only to deliver pre-formatted messages.

With no flags, *sendmail* reads its standard input up to a control-D or a line with a single dot and sends a copy of the letter found there to all of the addresses listed. It determines the network to use based on the syntax and contents of the addresses.

Local addresses are looked up in a file and aliased appropriately. Aliasing can be prevented by preceding the address with a backslash. Normally the sender is not included in any alias expansions, e.g., if 'john' sends to 'group', and 'group' includes 'john' in the expansion, then the letter will not be delivered to 'john'.

Flags are:

–ba	Go into ARPANET mode. All input lines must end with a CR-LF, and all messages will be generated with a CR-LF at the end. Also, the "From:" and "Sender:" fields are examined for the name of the sender.
–bd	Run as a daemon. This requires Berkeley IPC.
–bi	Initialize the alias database.
–bm	Deliver mail in the usual way (default).
–bp	Print a listing of the queue.
–bs	Use the SMTP protocol as described in RFC821. This flag implies all the operations of the **–ba** flag that are compatible with SMTP.
–bt	Run in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.
–bv	Verify names only – do not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.
–bz	Create the configuration freeze file.
–C*file*	Use alternate configuration file.
–d*X*	Set debugging value to *X*.

−F*fullname*	Set the full name of the sender.
−f*name*	Sets the name of the "from" person (i.e., the sender of the mail). **−f** can only be used by the special users *root, daemon,* and *network,* or if the person you are trying to become is the same as the person you are.
−h*N*	Set the hop count to *N*. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop.
−n	Don't do aliasing.
−o*x value*	Set option *x* to the specified *value*. Options are described below.
−q[*time***]**	Processed saved messages in the queue at given intervals. If is omitted, process the queue once. is given as a tagged number, with 's' being seconds, 'm' being minutes, 'h' being hours, 'd' being days, and 'w' being weeks. For example, "−q1h30m" or "−q90m" would both set the timeout to one hour thirty minutes.
−r*name*	An alternate and obsolete form of the **−f** flag.
−t	Read message for recipients. To:, Cc:, and Bcc: lines will be scanned for people to send to. The Bcc: line will be deleted before transmission. Any addresses in the argument list will be suppressed.
−v	Go into verbose mode. Alias expansions will be announced, etc.

There are also a number of processing options that may be set. Normally these will only be used by a system administrator. Options may be set either on the command line using the **−o** flag or in the configuration file. These are described in detail in the *Installation and Operation Guide.* The options are:

A*file*	Use alternate alias file.
c	On mailers that are considered "expensive" to connect to, don't initiate immediate connection. This requires queueing.
d*x*	Set the delivery mode to *x*. Delivery modes are 'i' for interactive (synchronous) delivery, 'b' for background (asynchronous) delivery, and 'q' for queue only − i.e., actual delivery is done the next time the queue is run.
D	Try to automatically rebuild the alias database if necessary.
e*x*	Set error processing to mode *x*. Valid modes are 'm' to mail back the error message, 'w' to "write" back the error message (or mail it back if the sender is not logged in), 'p' to print the errors on the terminal (default), 'q' to throw away error messages (only exit status is returned), and 'e' to do special processing for the BerkNet. If the text of the message is not mailed back by modes 'm' or 'w' and if the sender is local to this machine, a copy of the message is appended to the file "dead.letter" in the sender's home directory.

F*mode*	The mode to use when creating temporary files.
f	Save UNIX-style From lines at the front of messages.
g*N*	The default group id to use when calling mailers.
H*file*	The SMTP help file.
i	Do not take dots on a line by themselves as a message terminator.
L*n*	The log level.
m	Send to "me" (the sender) also if I am in an alias expansion.
o	If set, this message may have old style headers. If not set, this message is guaranteed to have new style headers (i.e., commas instead of spaces between addresses). If set, an adaptive algorithm is used that will correctly determine the header format in most cases.
Q*queuedir*	Select the directory in which to queue messages.
r*timeout*	The timeout on reads; if none is set, *sendmail* will wait forever for a mailer.
S*file*	Save statistics in the named file.
s	Always instantiate the queue file, even under circumstances where it is not strictly necessary.
T*time*	Set the timeout on messages in the queue to the specified time. After sitting in the queue for this amount of time, they will be returned to the sender. The default is three days.
t*stz,dtz*	Set the name of the time zone.
u*N*	Set the default user id for mailers.

If the first character of the user name is a vertical bar, the rest of the user name is used as the name of a program to pipe the mail to. It may be necessary to quote the name of the user to keep *sendmail* from suppressing the blanks from between arguments.

Sendmail returns an exit status describing what it did. The codes are defined in *<sysexits.h>*

EX_OK	Successful completion on all addresses.
EX_NOUSER	User name not recognized.
EX_UNAVAILABLE	Catchall meaning necessary resources were not available.
EX_SYNTAX	Syntax error in address.
EX_SOFTWARE	Internal software error, including bad arguments.
EX_OSERR	Temporary operating system error, such as "cannot fork".
EX_NOHOST	Host name not recognized.
EX_TEMPFAIL	Message could not be sent immediately, but was queued.

If invoked as *newaliases, sendmail* will rebuild the alias database. If invoked as *mailq, sendmail* will print the contents of the mail queue.

FILES

Except for /usr/lib/sendmail.cf, these pathnames are all specified in /usr/lib/sendmail.cf. Thus, these values are only approximations.

/usr/lib/aliases	raw data for alias names
/usr/lib/aliases.pag	
/usr/lib/aliases.dir	data base of alias names
/usr/lib/sendmail.cf	configuration file
/usr/lib/sendmail.fc	frozen configuration
/usr/lib/sendmail.hf	help file
/usr/lib/sendmail.st	collected statistics
/usr/bin/uux	to deliver uucp mail
/usr/net/bin/v6mail	to deliver local mail
/usr/net/bin/sendberkmail	to deliver Berknet mail
/usr/lib/mailers/arpa	to deliver ARPANET mail
/usr/spool/mqueue/*	temp files

SEE ALSO

biff(1), binmail(1), mail(1), aliases(5), sendmail.cf(5), rmail(1), mailaddr(7);
DARPA Internet Request For Comments RFC819, RFC821, RFC822;
Sendmail – An Internetwork Mail Router;
Sendmail Installation and Operation Guide.

RESTRICTIONS

Sendmail converts blanks in addresses to dots. This is incorrect according to the old ARPANET mail protocol RFC733 (NIC 41952), but is consistent with the new protocols (RFC822).

STATUS

SENDMAIL(8) is supported by Digital Equipment Corporation.

NAME

setifaddr – set network interface address

SYOPNSIS

/etc/setifaddr interface address

DESCRIPTION

Setifaddr assigns a network address to a network interface. It must be used at boot time to define the network address of each interface present on a machine. It may also be used at a later time to redefine an interface's address. The *interface* parameter is a string of the form "name unit", e.g. "en0", while the address is either a host name present in the host name data base, *hosts*(5), or a DARPA Internet address expressed in the Internet standard "dot notation".

DIAGNOSTICS

Messages indicating the specified interface does not exit, the requested address is unknown.

SEE ALSO

rc(8), intro(4N)

STATUS

SETIFADDR(8C) currently is not supported by Digital Equipment Corporation.

NAME

shutdown – close down the system at a given time

SYNTAX

/etc/shutdown [**–k**] [**–r**] [**–h**] time [warning-message ...]

DESCRIPTION

Shutdown provides an automated shutdown procedure which a super-user can use to notify users nicely when the system is shutting down, saving them from system administrators, hackers, and gurus, who would otherwise not bother with niceties.

Time is the time at which *shutdown* will bring the system down and may be the word **now** (indicating an immediate shutdown) or specify a future time in one of two formats: +number and hour:min. The first form brings the system down in *number* minutes and the second brings the system down at the time of day indicated (as a 24–hour clock).

At intervals which get closer together as apocalypse approaches, warning messages are displayed at the terminals of all users on the system. Five minutes before shutdown, or immediately if shutdown is in less than 5 minutes, logins are disabled by creating /etc/nologin and writing a message there. If this file exists when a user attempts to log in, *login*(1) prints its contents and exits. The file is removed just before *shutdown* exits.

At shutdown time a message is written in the file /usr/adm/shutdownlog, containing the time of shutdown, who ran shutdown and the reason. Then a terminate signal is sent at *init* to bring the system down to single-user state. Alternatively, if **–r, –h,** or **–k** was used, then *shutdown* will exec *reboot*(8), *halt*(8), or avoid shutting the system down (respectively). (If it isn't obvious, **–k** is to make people *think* the system is going down!)

The time of the shutdown and the warning message are placed in /etc/nologin and should be used to inform the users about when the system will be back up and why it is going down (or anything else).

FILES

/etc/nologin tells login not to let anyone log in
/usr/adm/shutdownlog log file for successful shutdowns.

SEE ALSO

login(1), reboot(8)

RESTRICTIONS

Only allows you to kill the system between now and 23:59 if you use the absolute time for shutdown.

STATUS

SHUTDOWN(8) currently is not supported by Digital Equipment Corporation.

NAME

sticky – executable files with persistent text

DESCRIPTION

While the 'sticky bit', mode 01000 (see *chmod*(2)), is set on a sharable executable file, the text of that file will not be removed from the system swap area. Thus the file does not have to be fetched from the file system upon each execution. As long as a copy remains in the swap area, the original text cannot be overwritten in the file system, nor can the file be deleted. (Directory entries can be removed so long as one link remains.)

Sharable files are made by the **−n** and **−z** options of *ld*(1).

To replace a sticky file that has been used do: (1) Clear the sticky bit with *chmod*(1). (2) Execute the old program to flush the swapped copy. This can be done safely even if others are using it. (3) Overwrite the sticky file. If the file is being executed by any process, writing will be prevented; it suffices to simply remove the file and then rewrite it, being careful to reset the owner and mode with *chmod* and *chown*(2). (4) Set the sticky bit again.

Only the super-user can set the sticky bit.

RESTRICTIONS

Is largely unnecessary on the VAX; matters only for large programs that will page heavily to start, since text pages are normally cached incore as long as possible after all instances of a text image exit.

STATUS

STICKY(8) currently is not supported by Digital Equipment Corporation.

NAME

swapon – specify additional device for paging and swapping

SYNTAX

/etc/swapon –a
/etc/swapon name ...

DESCRIPTION

Swapon is used to specify additional devices on which paging and swapping are to take place. The system begins by swapping and paging on only a single device so that only one disk is required at bootstrap time. Calls to *swapon* normally occur in the system multi-user initialization file */etc/rc* making all swap devices available, so that the paging and swapping activity is interleaved across several devices.

Normally, the **–a** argument is given, causing all devices marked as "sw" swap devices in **/etc/fstab** to be made available.

The second form gives individual block devices as given in the system swap configuration table. The call makes only this space available to the system for swap allocation.

SEE ALSO

swapon(2), init(8)

FILES

/dev/[ru][pk]?b normal paging devices

RESTRICTIONS

There is no way to stop paging and swapping on a device. It is therefore not possible to make use of devices which may be dismounted during system operation.

STATUS

SWAPON(8) currently is not supported by Digital Equipment Corporation.

NAME

sync – update the super block

SYNTAX

/etc/sync

DESCRIPTION

Sync executes the *sync* system primitive. *Sync* can be called to insure all disk writes have been completed before the processor is halted in a way not suitably done by *reboot*(8) or *halt*(8).

See *sync*(2) for details on the system primitive.

SEE ALSO

sync(2), fsync(2), halt(8), reboot(8), update(8)

STATUS

SYNC(8) is supported by Digital Equipment Corporation.

NAME

syslog – log systems messages

SYNTAX

/etc/syslog [**−m***N*] [**−f***name*] [**−d**]

DESCRIPTION

Syslog reads a datagram socket and logs each line it reads into a set of files described by the configuration file /etc/syslog.conf. *Syslog* configures when it starts up and whenever it receives a hangup signal.

Each message is one line. A message can contain a priority code, marked by a digit in angle braces at the beginning of the line. Priorities are defined in <*syslog.h*>, as follows:

LOG_ALERT this priority should essentially never be used. It applies only to messages that are so important that every user should be aware of them, e.g., a serious hardware failure.

LOG_SALERT messages of this priority should be issued only when immediate attention is needed by a qualified system person, e.g., when some valuable system resource dissappears. They get sent to a list of system people.

LOG_EMERG Emergency messages are not sent to users, but represent major conditions. An example might be hard disk failures. These could be logged in a separate file so that critical conditions could be easily scanned.

LOG_ERR these represent error conditions, such as soft disk failures, etc.

LOG_CRIT such messages contain critical information, but which can not be classed as errors, for example, 'su' attempts. Messages of this priority and higher are typically logged on the system console.

LOG_WARNING issued when an abnormal condition has been detected, but recovery can take place.

LOG_NOTICE something that falls in the class of "important information"; this class is informational but important enough that you don't want to throw it away casually. Messages without any priority assigned to them are typically mapped into this priority.

LOG_INFO information level messages. These messages could be thrown away without problems, but should be included if you want to keep a close watch on your system.

LOG_DEBUG it may be useful to log certain debugging information. Normally this will be thrown away.

It is expected that the kernel will not log anything below LOG_ERR priority.

The configuration file is in two sections separated by a blank line. The first section defines files that *syslog* will log into. Each line contains a single digit which defines the lowest priority (highest numbered priority) that this file will receive, an optional asterisk which

guarantees that something gets output at least every 20 minutes, and a pathname. The second part of the file contains a list of users that will be informed on SALERT level messages. For example, the configuration file:

```
5*/dev/console
8/usr/spool/adm/syslog
3/usr/adm/critical

eric
kridle
kalash
```

logs all messages of priority 5 or higher onto the system console, including timing marks every 20 minutes; all messages of priority 8 or higher into the file /usr/spool/adm/syslog; and all messages of priority 3 or higher into /usr/adm/critical. The users "eric", "kridle", and "kalash" will be informed on any subalert messages.

The flags are:

−m Set the mark interval to *N* (default 20 minutes).

−f Specify an alternate configuration file.

−d Turn on debugging (if compiled in).

To bring *syslog* down, it should be sent a terminate signal. It logs that it is going down and then waits approximately 30 seconds for any additional messages to come in.

There are some special messages that cause control functions. "<*>N" sets the default message priority to *N*. "<$>" causes *syslog* to reconfigure (equivalent to a hangup signal). This can be used in a shell file run automatically early in the morning to truncate the log.

Syslog creates the file /etc/syslog.pid if possible containing a single line with its process id. This can be used to kill or reconfigure *syslog*.

FILES

/etc/syslog.conf − the configuration file
/etc/syslog.pid − the process id

SEE ALSO

syslog(3)

RESTRICTIONS

LOG_ALERT and LOG_SUBALERT messages should only be allowed to privileged programs.

Actually, *syslog* is not clever enough to deal with kernel error messages in the current implementation.

STATUS

SYSLOG(8) currently is not supported by Digital Equipment Corporation.

NAME

telnetd – DARPA TELNET protocol server

SYNTAX

/etc/telnetd [**–d**] [*port*]

DESCRIPTION

Telnetd is a server which supports the DARPA standard TELNET virtual terminal protocol. The TELNET server operates at the port indicated in the "telnet" service description; see *services*(5). This port number may be overridden (for debugging purposes) by specifying a port number on the command line. If the **–d** option is specified, each socket created by *telnetd* will have debugging enabled (see SO_DEBUG in *socket*(2)).

Telnetd operates by allocating a pseudo-terminal device (see *pty*(4)) for a client, then creating a login process which has the slave side of the pseudo-terminal as **stdin**, **stdout**, and **stderr**. *Telnetd* manipulates the master side of the pseudo terminal, implementing the TELNET protocol and passing characters between the client and login process.

When a TELNET session is started up, *telnetd* sends a TELNET option to the client side indicating a willingness to do "remote echo" of characters. The pseudo terminal allocated to the client is configured to operate in "cooked" mode, and with XTABS and CRMOD enabled (see *tty*(4)). Aside from this initial setup, the only mode changes *telnetd* will carry out are those required for echoing characters at the client side of the connection.

Telnetd supports binary mode, and most of the common TELNET options, but does not, for instance, support timing marks. Consult the source code for an exact list of which options are not implemented.

SEE ALSO

telnet(1C)

STATUS

TELNETD(8C) currently is not supported by Digital Equipment Corporation.

NAME

tftpd – DARPA Trivial File Transfer Protocol server

SYNTAX

/etc/tftpd [**–d**] [*port*]

DESCRIPTION

Tftpd is a server which supports the DARPA Trivial File Transfer Protocol. The TFTP server operates at the port indicated in the "tftp" service description; see *services*(5). This port number may be overridden (for debugging purposes) by specifying a port number on the command line. If the **–d** option is specified, each socket created by *tftpd* will have debugging enabled (see SO_DEBUG in *socket*(2)).

The use of *tftp* does not require an account or password on the remote system. Due to the lack of authentication information, *tftpd* will allow only publicly readable files to be accessed. Note that this extends the concept of "public" to include all users on all hosts that can be reached through the network; this may not be appropriate on all systems, and its implications should be considered before enabling tftp service.

SEE ALSO

tftp(1C)

RESTRICTIONS

This server is known only to be self consistent (i.e. it operates with the user TFTP program, *tftp*(1C)). Due to the unreliability of the transport protocol (UDP) and the scarcity of TFTP implementations, it is uncertain whether it really works.

The search permissions of the directories leading to the files accessed are not checked.

STATUS

TFTPD(8C) currently is not supported by Digital Equipment Corporation.

NAME

trpt – transliterate protocol trace

SYNTAX

trpt [**–a**] [**–s**] [**–t**] [**–j**] [**–p** hex-address] [system [core]]

DESCRIPTION

Trpt interrogates the buffer of TCP trace records created when a socket is marked for "debugging" (see *setsockopt*(2)), and prints a readable description of these records. When no options are supplied, *trpt* prints all the trace records found in the system grouped according to TCP connection protocol control block (PCB). The following options may be used to alter this behavior.

–s in addition to the normal output, print a detailed description of the packet sequencing information,

–t in addition to the normal output, print the values for all timers at each point in the trace,

–j just give a list of the protocol control block addresses for which there are trace records,

–p show only trace records associated with the protocol control block who's address follows,

–a in addition to the normal output, print the values of the source and destination addresses for each packet recorded.

The recommended use of *trpt* is as follows. Isolate the problem and enable debugging on the socket(s) involved in the connection. Find the address of the protocol control blocks associated with the sockets using the **–A** option to *netstat*(1). Then run *trpt* with the **–p** option, supplying the associated protocol control block addresses. If there are many sockets using the debugging option, the **–j** option may be useful in checking to see if any trace records are present for the socket in question.

If debugging is being performed on a system or core file other than the default, the last two arguments may be used to supplant the defaults.

FILES

/vmunix
/dev/kmem

SEE ALSO

setsockopt(2), netstat(1)

DIAGNOSTICS

"no namelist" when the system image doesn't contain the proper symbols to find the trace buffer; others which should be self explanatory.

STATUS

TRPT(8C) currently is not supported by Digital Equipment Corporation.

NAME

tunefs – tune up an existing file system

SYNTAX

/etc/tunefs *tuneup-options special|filesys*

DESCRIPTION

Tunefs is designed to change the dynamic parameters of a file system which affect the layout policies. The parameters which are to be changed are indicated by the flags given below:

–a maxcontig

This specifies the maximum number of contiguous blocks that will be laid out before forcing a rotational delay (see –d below). The default value is one, since most device drivers require an interrupt per disk transfer. Device drivers that can chain several buffers together in a single transfer should set this to the maximum chain length.

–d rotdelay

This specifies the expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to decide how much rotational spacing to place between successive blocks in a file.

–e maxbpg

This indicates the maximum number of blocks any single file can allocate out of a cylinder group before it is forced to begin allocating blocks from another cylinder group. Typically this value is set to about one quarter of the total blocks in a cylinder group. The intent is to prevent any single file from using up all the blocks in a single cylinder group, thus degrading access times for all files subsequently allocated in that cylinder group. The effect of this limit is to cause big files to do long seeks more frequently than if they were allowed to allocate all the blocks in a cylinder group before seeking elsewhere. For file systems with exclusively large files, this parameter should be set higher.

–m minfree

This value specifies the percentage of space held back from normal users; the minimum free space threshold. The default value used is 10%. This value can be set to zero, however up to a factor of three in throughput will be lost over the performance obtained at a 10% threshold. Note that if the value is raised above the current usage level, users will be unable to allocate files until enough files have been deleted to get under the higher threshold.

SEE ALSO

fs(5), newfs(8), mkfs(8)

McKusick, Joy, Leffler; "A Fast File System for Unix", Computer Systems Research Group, Dept of EECS, Berkeley, CA 94720; TR #7, September 1982.

RESTRICTIONS

This program should work on mounted and active file systems. Because the super-block is not kept in the buffer cache, the program will only take effect if it is run on dismounted file systems. (if run on the root file system, the system must be rebooted)

STATUS

TUNEFS(8) currently is not supported by Digital Equipment Corporation.

NAME

update – periodically update the super block

SYNTAX

/etc/update

DESCRIPTION

update is a program that executes the *sync*(2) primitive every 30 seconds. This insures that the file system is fairly up to date in case of a crash. This command should not be executed directly. Rather, it should be executed out of the initialization shell command file.

SEE ALSO

sync(2), sync(8), init(8), rc(8)

RESTRICTIONS

With *update* running, if the CPU is halted just as the *sync* is executed, a file system can be damaged. This is partially due to DEC hardware that writes zeros when NPR requests fail. A fix would be to have *sync*(8) temporarily increment the system time by at least 30 seconds to trigger the execution of *update*. This would give 30 seconds grace to halt the CPU.

STATUS

update(8) is supported by Digital Equipment Corporation.

NAME

uucompact, uumkspool, uurespool – uucp administrative utilities

SYNTAX

uucompact -s*system*

uumkspool system(s)

uurespool [-t#]

DESCRIPTION

All three of these commands are located in /usr/lib/uucp.

Uucompact.

Uucompact compacts uucp system spool directories and associated subdirectories. If *system* is **ALL**, then all existing uucp system spool directories are compacted. Otherwise, only the specified *system* spool directory is compacted. If no *system* is specified, **/usr/spool/uucp/sys** is compacted. If *uucompact* is stopped before it is finished, it can be restarted without reprocessing directories. *Uucompact* continues processing where it left off during it's previous instantiation.

Uumkspool.

For each of the specified systems *uumkspool* makes a per system spool directory and associated subdirectories. For example, if *system* is **mk3** and if the local system name is **decvax**, the following directories are created:

/usr/spool/uucp/sys/mk3
/usr/spool/uucp/sys/mk3/C.
/usr/spool/uucp/sys/mk3/X.
/usr/spool/uucp/sys/mk3/D.
/usr/spool/uucp/sys/mk3/D.decvax
/usr/spool/uucp/sys/mk3/D.decvaxX

Uurespool.

Uurespool moves files from old spool directories to new spool directories. There are at least two instances where one will want to move spool files to a new spool directory.

1. Installing the current version of *uucp*. Because the structure of the spool directories has changed from older versions of *uucp*, it is necessary to respool old spooled files to new spool directories.

2. Creating a new per system spool directory. In this case, it is necessary to move files from
/usr/spool/uucp/sys/DEFAULT to the new spool directories. To ease this task, *uurespool* moves files that have been spooled in one of 4 formats and respools them under the new spooling structure. The format is specifed by the **-t#** option. # can be one of the following:

   1) original - all files are in /usr/spool/uucp

2) split spool - contains the following subdirectories: C., X., D., D.local, D.localX

3) modified split spool - contains all subdirectories listed in 2) plus: STST., TM., C./OTHERS

4) used when a new system directory has been created and spool files must be moved from the DEFAULT directory to the new system directory.

FILES

/usr/spool/uucp - spool directory

SEE ALSO

uucp (1C), uux(1C), mail(1)

STATUS

UUAIDS(8C) is supported by Digital Equipment Corporation.

NAME

uuclean – uucp spool directory clean-up

SYNTAX

uuclean [option] ...

DESCRIPTION

Uuclean scans the spool directory for files with the specified prefix and deletes all those which are older than the specified number of hours.

The following options are available.

-p*pre* Scan for files with *pre* as the file prefix. Up to 10 **-p** arguments may be specified. A **-p** without any *pre* following will cause all files older than the specified time to be deleted.

-n*time* Delete all files whose age is more than *time* (default is 72 hours) and that have the specified *pre* as their file prefix.

-m Send mail to the owner of the file when it is deleted.

-s*system*
Delete files in all directories that are subdirectories of the per system spool directory that exists for *system*. If **ALL** is specified, then all system directories are processed.

-d*directory*
Delete files that reside in the named *directory*. The **-s** option overides the **-d** option.

This program will typically be started by *cron*(8). In earlier versions, a deleted work file (C.file) would result in mail to the owner of the work file, regardless of the **-m** option. Now, notification of deleted work files is sent to the user id "uucp". If the **-m** option is used, mail is also sent to the owner.

FILES

/usr/lib/uucp directory with commands used by uuclean internally

SEE ALSO

uucp(1C), uux(1C)

STATUS

UUCLEAN(8C) is supported by Digital Equipment Corporation.

NAME

uumonitor – monitor the UUCP system

SYNTAX

uumonitor

DESCRIPTION

Uumonitor displays a synopsis in tabular format of the current UUCP status. The format of each line is as follows:

system name #C #X most recent status CNT:# time

where

system name
is the remote system for which the entry applies.

#C is the number of C.files queued for the remote system.

#X is the number of requests for remote execution from the remote system.

most recent status
is the result of the most recent attempt to connect to the remote system.

CNT:# is the number of times that a failure to login to the remote system has occured. This does NOT include the number failed dial attempts.

time is the time of the last status entry was made for this system.

This command is helpful for detecting systems that have backlogs, that have "gone away" for awhile, that have changed phone numbers, etc. The CNT: field is useful for detecting a system whose login/passwd has changed. If the CNT: field gets larger than the maximum allowable failures (currently 20), no further attempts to connect to this system are made. If the number of C.files queued starts getting unusually large (depending on the system "large" could mean anywhere from 100-1000), action should be taken to determine the cause of the backlog.

SEE ALSO

uucp(1C)
UUCP Implementation Guide
UUCP Installation and Administration Guide.

STATUS

UUMONITOR(8C) is supported by Digital Equipment Corporation.

NAME

vipw – edit the password file

SYNTAX

vipw

DESCRIPTION

vipw edits the password file while setting the appropriate locks and doing any necessary processing after the password file is unlocked. If the password file is already being edited, then you will be told to try again later. The *vi* editor will be used unless the environment variable EDITOR indicates an alternate editor. *vipw* performs a number of consistency checks on the password entry for *root*, and does not allow a password file with a corrupted root entry to be installed.

SEE ALSO

chfn(1), chsh(1), passwd(1), passwd(5), adduser(8)

FILES

/etc/ptmp

STATUS

VIPW(8) currently is not supported by Digital Equipment Corporation.

HOW TO ORDER ADDITIONAL DOCUMENTATION

DIRECT TELEPHONE ORDERS

In Continental USA and Puerto Rico call **800–258–1710**

In Canada call **800–267–6146**

In New Hampshire, Alaska or Hawaii call **603–884–6660**

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.
940 Belfast Road
Ottawa, Ontario, Canada K1G 4C2
Attn: A&SG Business Manager

INTERNATIONAL

DIGITAL EQUIPMENT CORPORATION
A&SG 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, Northboro, Massachusetts 01532

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

ULTRIX - 32
Programmer's Manual
AA-BG56A-TE

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

Documentation Manager
ULTRIX-32™ Documentation Group
MK02-1/H10
Continental Blvd.
Merrimack, N.H.
03054

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line

Notes:

Notes:

Notes:

Notes:

Notes:

Notes:

Notes:

Notes:

digital™