BSD Unix 2.11 man entries
6/9/2019 - The ShadowTron Blog

Generated using the simh PDP-11/70 emulator with the PiDP11 Front Panel

PiDP11 - https://obsolescence.wixsite.com/obsolescence/pidp-11
SimH - http://simh.trailing-edge.com/
ShadowTronBlog - https://www.youtube.com/channel/UCtUiwjYcRS_u6Egc8iTkHNg
http://shadowtron.com
shadowtronblog@gmail.com

Manual Area covered
====== ==========
1 Commands and Application Programs
2 System Calls
3 C Library Subroutines
3F Fortran Library
4 Special Files
5 File Formats
6 Games
7 Miscellaneous
--> 8 System Maintenance

```
**************************************
**** Manual 8 - System Maintenance ****
**************************************
    Intro               introduction to system maintenance and operation commands
    ac                  login accounting
    adduser             procedure for adding new users
    arff                archiver and copier for floppy
    arp                 address resolution display and control
    bad144              read/write dec standard 144 bad sector information
    badsect             create files to contain bad sectors
    bugfiler            file bug reports in folders automatically
    catman              create the cat files for the manual
    chown               change owner
    clri                clear i-node
    comsat              biff server
    config              build system configuration files
    crash               what happens when the system crashes
    cron                clock daemon
    dcheck              file system directory consistency check
    diskpart            calculate default disk partition sizes
    dmesg               collect system diagnostic messages to form error log
    drtest              standalone disk test program
    dump                incremental file system dump
    dumpfs              dump file system information
    edquota             edit user quotas
    fastboot            reboot/halt the system without checking the disks
    fingerd             remote user information server
    format              how to format disk packs
    fsck                file system consistency check and interactive repair
    ftpd                DARPA Internet File Transfer Protocol server
    gettable            get NIC format host tables from a host
    getty               set terminal mode
    halt                stop the processor
    htable              convert NIC standard format host tables
    icheck              file system storage consistency check
    ifconfig            configure network interface parameters
    implog              IMP log interpreter
    implogd             IMP logger process
    inetd               internet "super\server"
    init                process control initialization
    kgmon               generate a dump of the operating system's profile buffers
    lpc                 line printer control program
    lpd                 line printer daemon
    makedev             make system special files
    makekey             generate encryption key
    mkfs                construct a file system
    mkhosts             generate hashed host table
    mklost+found        make a lost+found directory for fsck
    mknod               build special file
    mkpasswd            generate hashed password table
    mkproto             construct a prototype file system
    mount               mount and dismount file system
    named               Internet domain name server
    ncheck              generate names from i-numbers
    newfs               construct a new file system
    pac                 printer/plotter accounting information
    ping                send ICMP ECHO_REQUEST packets to network hosts
    pstat               print system facts
    quot                summarize file system ownership
    quotacheck          file system quota consistency checker
```

quotaon	turn file system quotas on and off
rc	command script for auto-reboot and daemons
rdump	file system dump across the network
reboot	UNIX bootstrapping procedures
renice	alter priority of running processes
repquota	summarize quotas for a file system
restore	incremental file system restore
rexecd	remote execution server
rlogind	remote login server
rmt	remote magtape protocol module
route	manually manipulate the routing tables
routed	network routing daemon
rrestore	restore a file system dump across the network
rshd	remote shell server
rwhod	system status server
rxformat	format floppy disks
sa	system accounting
savecore	save a core dump of the operating system
sendmail	send mail over the internet
shutdown	close down the system at a given time
slattach	attach serial lines as network interfaces
sticky	persistent text and append-only directories
swapon	specify additional device for paging and swapping
sync	update the super block
syslogd	log systems messages
talkd	remote user communication server
telnetd	DARPA TELNET protocol server
tftpd	DARPA Trivial File Transfer Protocol server
timed	time server daemon
timedc	timed control program
trpt	transliterate protocol trace
trsp	transliterate sequenced packet protocol trace
tunefs	tune up an existing file system
update	periodically update the super block
uucico	transfer files queued by uucp or uux
uuclean	uucp spool directory clean-up
uupoll	poll a remote UUCP site
uusnap	show snapshot of the UUCP system
uuxqt	UUCP execution file interpreter
vipw	edit the password file
XNSrouted	NS Routing Information Protocol daemon

NAME

intro - introduction to system maintenance and operation commands

DESCRIPTION

This section contains information related to system operation and maintenance. It describes commands used to create new file systems, newfs, verify the integrity of the file systems, fsck, control disk usage, edquota, maintain system backups, dump, and recover files when disks die an untimely death, restore. The section format should be consulted when formatting disk packs. Network related services are distinguished as 8C. The section crash should be consulted in understanding how to interpret system crash dumps.

NAME

ac - login accounting

SYNOPSIS

/usr/sbin/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).

NAME

adduser - procedure for adding new users

DESCRIPTION

A new user must choose a login name, which must not already appear in /etc/passwdor /etc/aliases. It must also not begin with the hyphen (``-'') character. It is strongly recommended that it be all lower-case, and not contain the dot (``.'') character, as that tends to confuse mailers. An account can be added by editing a line into the passwd file; this must be done with the password file locked e.g. by using chpass(1) or vipw(8).

A new user is given a group and user id. Login's and user id's should be unique across the system, and often across a group of systems, since they are used to control file access. Typically, users working on similar projects will be put in the same groups. At the University of California, Berkeley, we have groups for system staff, faculty, graduate students, and special groups for large projects.

A skeletal account for a new user "ernie" might look like:

```
ernie::25:30::0:0:Ernie Kovacs,508 Evans
Hall,x7925,642-8202:/a/users/ernie:/bin/csh
```

For a description of each of these fields, see passwd(5).

It is useful to give new users some help in getting started, supplying them with a few skeletal files such as .profile if they use "/bin/sh", or .cshrc and .login if they use "/bin/csh". The directory "/usr/skel" contains skeletal definitions of such files. New users should be given copies of these files which, for instance, use tset(1) automatically at each login.

FILES

/etc/master.passwd	user database
/usr/skel	skeletal login directory

SEE ALSO

chpass(1), finger(1), passwd(1), aliases(5), passwd(5), mkpasswd(8), vipw(8)

BUGS

User information should (and eventually will) be stored elsewhere.

NAME

autoconfig - configure the running system to the hardware

SYNOPSIS

autoconfig [-i ifile] [-n nfile] [-k kfile] [-v] [-d] [-c]

DESCRIPTION

Autoconfig is called by init(8) to configure the currently running system. Init checks the exit status of autoconfig to determine if the configuration was successful. Autoconfig reads the device table /etc/dtab for a list of devices which may be on the system. It first verifies that the kernel has an attach routine for each device (and therefore has a device handler) and that the kernel has a probe routine. It then checks each of these devices to see if it is present, and if it is, attempts to make it interrupt (if possible) to verify that the interrupt vector is correct. The interrupt vector is checked to see that it has not previously been used. An interrupt through any of the device's consecutive vectors is sufficient.

Devices which use programmable vectors (MSCP and TMSCP) are permitted to have a value of 0 in the dtab vector field. This special value tells autoconfig to call the kernel's get next available vector routine and assign that to the device. For programmable vector devices if the dtab vector field is non 0 then the value specified in the dtab file is used. In both cases the driver is called at its xxVec() routine with the vector being assigned to the device.

If the address and vector are correct, it then attaches the device by passing the address and unit number to the kernel's attach routine and setting up the interrupt vector according to the interrupt handlers and priority listed in the device table. If the unit number is given as a '?' in the device table, it will be assigned the next available unit number if the device exists. If the device is not present or the vector is incorrect, and if the unit number was specified (not a '?'), then the kernel is notified that that unit is not present, preventing accesses to a nonexistent device address.

There are only a few flags which are mostly useful for debugging but for completeness, here they are.

-i ifile Use ifile instead of /etc/dtab as the device table.

-n nfile Use nfile instead of /unix for finding the namelist of the currently running kernel.

-k kfile The file kfile should be used instead of /dev/kmem

to alter and read kernel memory.

-v Verbose output, indicates reason for rejecting any device in the device table. Normally only attached devices are reported.

-c Report error messages for devices skipped because of problems with their interrupt vectors.

-d Turn on debugging mode. Shows many gory details of autoconfig's internal processing.

BUGS

Devices of the same type must be listed with ascending unit numbers or with wildcards.

Disks that could be root devices must have their addresses and vectors initialized in the kernel; the kernel uses a root attach entry in the block device switch to allow disk drivers to do any probes necessary before autoconfiguration.

Must be run only by init(8). There is a flag set in the kernel that autoconfig has already run, running autoconfig a second time results in the error:

"namelist doesn't match running kernel."
Autoconfig attempts to open /dev/kmem for write. If the kernel is in securelevel 1 or higher the open of /dev/kmem will fail.

FILES

/etc/dtab device table
/unix
/dev/kmem

SEE ALSO

ucall(2), nlist(3), dtab(5)

NAME

boot - 2.11BSD bootstrap proceedure

DESCRIPTION

The 2.11BSD system is started by a two-stage process. The first is a primary bootstrap (limited to 512 bytes) which is able to read in relatively small stand-alone programs; the second (called boot) is used to read in the system itself.

The primary bootstrap must reside in block zero of the boot device (the disklabel resides in block one). It can be read in and started by standard ROM cold boot routines or, if necessary, by keying in a small startup routine. The primary bootstrap is capable of loading only type 0407 executable files (impure (non-shared), non-separate I&D.) Copies of the block zero bootstraps are kept in the directory /mdec. Disklabel(8) is normally used to place a copy of the appropriate bootstrap in block zero of new file systems.

The primary bootstrap loads boot from the file system that starts at block 0 of the drive specified to the boot ROM. Normally the boot device is automatically used as the root filesystem. This action can be overriden by specifying the -R command to boot. If boot is not found the system will hang as the primary boot spins in an endless loop trying to find boot. No diagnostic message results if the file cannot be found.

+ In an emergency, the bootstrap methods described in the paper Installing and Operating 2.11BSD can be used to boot from a distribution tape.

The secondary boot program, called boot, actually brings in the system. When read into location 0 and executed, boot sets up memory management, relocates itself into high memory, and types its name and a `:' on the console. If this is an automatic, unattended reboot, boot will use a default file specification for the installation, typing the file's name after the prompt. Otherwise, it reads a file specification from the console. Normal line editing characters can be used to make corrections while typing this (see below for file specification format). If only a carriage return is typed, a default name (/unix) will be used. Boot finds the [specified] file and loads it into memory location zero, sets up memory management as required, and calls the program by executing a `trap' instruction.

For the system to boot, /etc/init must exist and be executable; if it is not, the kernel will print a message to that effect and loop. Further, for a single user boot, the files /bin/sh and /dev/console must also exist and /bin/sh must be executable (if either of these is missing, init will attempt

multi-user operation). For a multi-user boot the file /etc/ttys must exist (if missing, init will attempt single user operation).

Init runs the autoconfig(8) program to probe for and initialize devices. Autoconfig only knows to look in /unix, thus if an alternate kernel name was specified none of the devices except /dev/console and the boot disk will be known.

If autoconfig problems are suspected (or if you are simply voyeuristic) the debug flag can be turned on by specifying -D to boot (see below).

When the system is running in single user mode, it starts a single user shell on the console which types a `#' prompt. After doing any file system checks and setting the date (date(1)) a multi-user system can be brought up by typing an EOT (control-d) in response to the `#' prompt.

Boot file specification format: The file specifications used with boot are of the form:

device(ctlr,unit,part)path [-aRrDs]

or

-bootcommand

where

device
is the type of the device to be searched;

ctlr is the controller number of the disk

unit is the unit number of the disk or tape;

part is the partition number of a filesystem on the specified disk or the tape file number if the device is a tape. The underlying device driver must support disklabels and a valid disklabel must be present if part is anything except 0.

path is the path name of a disk file to be loaded with all mount prefixes stripped off (path must be omitted for tape files.) Tape files are separated by single tape marks.

Flags to boot may be specified in either of two places. At the : prompt and after the file name. The options are:

-a Ask for a kernel name. This is present for symmetry

only because in order to specify this option you already have to be at the : prompt.

-D Turn on the autoconfig debug flag.

-R force the kernel to use its compiled in root device rather than adapting to the boot device.

-s tell init to enter single user state rather than bringing the system all the way up to multi-user mode. -r mount the root filesystem read-only. This is not currently supported by the kernel mostly because pipes are implemented in the filesystem.

Commands (-bootcommand) to boot are:

-bootflags N where N is a decimal number.

-bootflags flag where flag is from the list above.

-bootdebug N where N is a decimal number. This is a general purpose flag word used by boot and is not passed to the loaded program or kernel.

Device is one of the following

xp	RM02/03/05, RP04/05/06, DIVA, SI Eagle, CDC 9766, Fuji 160
rp	RP03
rk	RK05
hk	RK06/7
rl	RL01/2
si	RM05, CDC 9766
ra	RA60/80/81, RX50, RD51/52/53, RC25
ht	TU/TE16
tm	TU/TE10
ts	TS-11

The stand alone tape drive unit number is specially encoded to specify both unit number and tape density (BPI). Most tape subsystems either automatically adjust to tape density or have switches on the drives to force the density to a particular setting, but for those which don't the following density select mechanisms may be necessary. The ts only operates at 1600BPI, so there is no special unit density encoding. The ht will operate at either 800BPI or 1600BPI. Units 0 through 3 correspond to 800BPI, and 4 through 7 to 1600BPI on drives 0 through 3 respectively. The standard DEC tm only supports 800BPI (and hence can't be used with the standard distribution tape), but several widely used tm emulators support 1600BPI and even 6250BPI. Units 0 through 3 correspond to 800BPI, 4 through 7 to 1600BPI, and 8

through 11 to 6250BPI on drives 0 through 3 respectively.

For example, to boot a system from unit 0 on an RK07, type "hk(0,0)unix" to the boot prompt. The specification "ra(1,0)unix" indicates an MSCP disk, unit 1. The specification "ra(1,0,0)unix" indicates an MSCP disk, unit 0 but on controller 1. And finally the specification "ts(0,3)" would cause the fourth file on a tape threaded on `ts' tape drive 0 to be loaded and executed.

Cold boot loaders: The following programs to load and execute the primary bootstrap may be installed in read-only memories or manually keyed into main memory. Each program is position-independent but should be placed well above location 0 so it will not be overwritten. Each reads a block from the beginning of a device into core location zero. The octal words constituting the program are listed on the left.

RK (drive 0):

```
012700          mov     $rkda,r0
177412
005040          clr     -(r0)    / rkda cleared by start
010040          mov     r0,-(r0)
012740          mov     $5,-(r0)
000005
105710      1:  tstb    (r0)
002376          bge     1b
005007          clr     pc
```

RP (drive 0)

```
012700          mov     $rpmr,r0
176726
005040          clr     -(r0)
005040          clr     -(r0)
005040          clr     -(r0)
010040          mov     r0,-(r0)
012740          mov     $5,-(r0)
000005
105710      1:  tstb    (r0)
002376          bge     1b
005007          clr     pc
```

TM (drive 0):

```
012700          mov     $tmba,r0
172526
010040          mov     r0,-(r0)
012740          mov     $60003,-(r0)
060003
000777          br          .
```

FILES

/unix	system code
/boot	system bootstrap
/etc/init	system process dispatcher
/mdec/xxuboot	sector 0 boot blocks, xx is disk type

SEE ALSO

crash(8V), autoconfig(8), reboot(2), disklabel(8), fsck(8), init(8)

NAME

arff, flcopy - archiver and copier for floppy

SYNOPSIS

/usr/sbin/arff [key] [name ...]
/usr/sbin/flcopy [-h] [-tn]

DESCRIPTION

Arff saves and restores files on VAX console media (the console floppy on the VAX 11/780 and 785, the cassette on the 11/730, and the console RL02 on the 8600/8650). 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. The default options are correct for the RX01 floppy on the 780; for other console media, the f and m flags are required.

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. It must be used with TU58 or RL02 media.

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.

FILES

/dev/floppy or /dev/rrx??
floppy (in current directory)

SEE ALSO

crl(4), fl(4), rx(4), tu(4), rxformat(8)

AUTHORS

Keith Sklower, Richard Tuck

BUGS

Device errors are handled ungracefully.

NAME

arp - address resolution display and control

SYNOPSIS

```
arp hostname
arp -a [ vmunix ] [ kmem ]
arp -d hostname
arp -s hostname ether_addr [ temp ] [ pub ] [ trail ]
arp -f filename
```

DESCRIPTION

The arp program displays and modifies the Internet-to-Ethernet address translation tables used by the address resolution protocol (arp(4p)).

With no flags, the program displays the current ARP entry for hostname. The host may be specified by name or by number, using Internet dot notation. With the -a flag, the program displays all of the current ARP entries by reading the table from the file kmem (default /dev/kmem) based on the kernel file vmunix (default /vmunix).

With the -d flag, a super-user may delete an entry for the host called hostname.

The -s flag is given to create an ARP entry for the host called hostname with the Ethernet address ether_addr. The Ethernet address is given as six hex bytes separated by colons. The entry will be permanent unless the word temp is given in the command. If the word pub is given, the entry will be "published"; i.e., this system will act as an ARP server, responding to requests for hostname even though the host address is not its own. The word trail indicates that trailer encapsulations may be sent to this host.

The -f flag causes the file filename to be read and multiple entries to be set in the ARP tables. Entries in the file should be of the form

```
hostname ether_addr [ temp ] [ pub ] [ trail ]
```

with argument meanings as given above.

SEE ALSO

inet(3N), arp(4P), ifconfig(8C)

NAME

bad144 - read/write DEC standard 144 bad sector information

SYNOPSIS

bad144 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. Only the first of these copies is used.

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 can be supported. The position of the bad sector in the bad sector table determines which replacement sector it corresponds to.

The bad sector information and replacement sectors are conventionally only accessible through the ``h'' 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[MAXBAD];
};
```

Unused slots in the bt_bad array are filled with all bits set, a putatively illegal value. MAXBAD (in <sys/dkbad.h>) may be tuned locally to reduce the space required to hold the bad-sector file in memory. It may not be greater than 126, which uses the whole disk sector. Bad sectors past MAXBAD may be included by the formatter, but replacement sectors will not be used until MAXBAD is increased.

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.

New bad sectors can be added by running the standard DEC formatter in section ``bad.''

SEE ALSO

badsect(8)

BUGS

Not all drivers support bad-sector forwarding on the PDP-11.

It should be possible to both format disks on-line under UNIX and to change the bad sector information, marking new bad sectors, without running a standalone program.

The bootstrap drivers used to boot the system do not understand bad sectors or handle ECC errors. This means that none of these errors can occur when reading the file /unix to boot. Sector 0 of the disk drive and the file /boot in the root file system of that drive must also not have any of these errors in it.

The drivers that 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.

NAME

badsect - create files to contain bad sectors

SYNOPSIS

/sbin/badsect 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.

Adding a sector which is suddenly bad to the bad sector table currently requires the running of the standard DEC formatter, as UNIX does not supply formatters. 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 and change into it. Run badsect giving as argument all the bad sectors you wish to add. (The sector numbers should be given as physical disk sectors relative to the beginning of the file system, exactly as the system reports the 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 (after taking into account the filesystem's block size), creating a regular file whose first block address is the block containing bad sector and whose name is the bad sector number. The file has 0 length, but the check programs will still consider it to contain the block containing the sector. This has the pleasant effect that the sector is completely inaccessible to the containing file system since it is not available by accessing the file.

SEE ALSO

mknod(2), bad144(8), fsck(8)

BUGS

If both sectors which comprise a (1024 byte) disk block are bad, you should specify only one of them to badsect, as the blocks in the bad sector files actually cover both (bad) disk sectors.

On the PDP-11, only sector number less than 131072 may be specified on 1024-byte block filesystems, 65536 on 512-byte block filesystems. This is because only a short int is passed to the system from mknod.

NAME

bugfiler - file bug reports in folders automatically

SYNOPSIS

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 /etc/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; improperly formatted messages are filed in a folder named ``errors.'' Program maintainers can then log onto the system and check the summary file for bugs that pertain to them. Bug reports should be submitted in RFC822 format and aremust contain the following header lines to be properly indexed:

```
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>
```

The bug report may also be redistributed according to the index. If the file maildir/.redist exists, it is examined for a line beginning with the index name followed with a tab. The remainder of this line contains a comma-separated list of mail addresses which should receive copies of bugs with this index. The list may be continued onto multiple lines by ending each but the last with a backslash (`\').

FILES

/usr/sbin/sendmail	mail delivery program
/usr/libexec/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/.redist	the redistribution list
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)

BUGS

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.

NAME

catman - create the cat files for the manual

SYNOPSIS

/usr/sbin/catman [-p] [-n] [-w] [-M path] [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 whatis database.

If there is one parameter not starting with a `-', it is taken 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 the whatis database.

-p prints what would be done instead of doing it.

-w causes only the whatis database to be created. No manual reformatting is done.

-M updates manual pages located in the set of directories specified by path (/usr/man by default). Path has the form of a colon (`:') separated list of directory names, for example `/usr/local/man:/usr/man'. If the environment variable `MANPATH' is set, its value is used for the default path.

If the nroff source file contains only a line of the form `.so manx/yyy.x', a symbolic link is made in the catx directory to the appropriate preformatted manual page. This feature allows easy distribution of the preformatted manual pages among a group of associated machines with rdist(1). The nroff sources need not be distributed to all machines, thus saving the associated disk space. As an example, consider a local network with 5 machines, called mach1 through mach5. Suppose mach3 has the manual page nroff sources. Every night, mach3 runs catman via cron(8) and later runs rdist with a distfile that looks like:

```
MANSLAVES = ( mach1 mach2 mach4 mach5 )

MANUALS = (/usr/man/cat[1-8no] /usr/man/whatis)

${MANUALS} -> ${MANSLAVES}
      install -R;
      notify root;
```

FILES

/usr/man	default manual directory location
/usr/man/man?/*.*	raw (nroff input) manual sections
/usr/man/cat?/*.*	preformatted manual pages
/usr/man/whatis	whatis database
/usr/sbin/makewhatis	command script to make whatis database

SEE ALSO

man(1), cron(8), rdist(1)

BUGS

Acts oddly on nights with full moons.

The need for catman(8) is almost but not quite gone. Most of the manpages have been moved out of /usr/src/man into the sourcecode hierarchy. The recreation of the whatis database is the main use of catman now.

NAME

chown - change owner

SYNOPSIS

/usr/sbin/chown [-f -R] owner[.group] 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. An optional group may also be specified. The group may be either a decimal GID or a group name found in the group-ID 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.

When the -R option is given, chown recursively descends its directory arguments setting the specified owner. When symbolic links are encountered, their ownership is changed, but they are not traversed.

FILES

/etc/passwd

SEE ALSO

chgrp(1), chown(2), passwd(5), group(5)

NAME

comsat - biff server

SYNOPSIS

/usr/libexec/comsat

DESCRIPTION

Comsat is the server process which receives reports of incoming mail and notifies users if they have requested this service. Comsat receives messages on a datagram port associated with the ``biff'' service specification (see services(5) and inetd(8)). The one line messages are 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

/var/run/utmp to find out who's logged on and on what terminals

SEE ALSO

biff(1), inetd(8)

BUGS

The message header filtering is prone to error. The density of the information presented is near the theoretical minimum.

Users should be notified of mail which arrives on other machines than the one to which they are currently logged in.

The notification should appear in a separate window so it does not mess up the screen.

NAME

crash - what happens when the system crashes

DESCRIPTION

This section explains what happens when the system crashes and (very briefly) how to analyze crash dumps.

When the system crashes voluntarily it prints a message of the form

panic: why i gave up the ghost

on the console, takes a dump on a mass storage peripheral, and then invokes an automatic reboot procedure as described in reboot(8). Unless some unexpected inconsistency is encountered in the state of the file systems due to hardware or software failure, the system will then resume multi-user operations. If the automatic file system check fails, the file systems should be checked and repaired with fsck(8) before continuing.

The system has a large number of internal consistency checks; if one of these fails, then it will panic with a very short message indicating which one failed. In many instances, this will be the name of the routine which detected the error, or a two-word description of the inconsistency. A full understanding of most panic messages requires perusal of the source code for the system.

The most common cause of system failures is hardware failure, which can reflect itself in different ways. Here are the messages which are most likely, with some hints as to causes. Left unstated in all cases is the possibility that hardware or software error produced the message in some unexpected way.

iinit

This cryptic panic message results from a failure to mount the root filesystem during the bootstrap process. Either the root filesystem has been corrupted, or the system is attempting to use the wrong device as root filesystem. Usually, an alternate copy of the system binary or an alternate root filesystem can be used to bring up the system to investigate.

Can't exec /etc/init

This is not a panic message, as reboots are likely to be futile. Late in the bootstrap procedure, the system was unable to locate and execute the initialization process, init(8). The root filesystem is incorrect or has been corrupted, or the mode or type of /etc/init forbids execution.

hard IO err in swap
The system encountered an error trying to write to the swap device or an error in reading critical information from a disk drive. The offending disk should be fixed if it is broken or unreliable.

timeout table overflow
This really shouldn't be a panic, but until the data structure involved is made to be extensible, running out of entries causes a crash. If this happens, make the timeout table bigger. (NCALL in param.c)

trap type %o
An unexpected trap has occurred within the system; the trap types are:

0	bus error
1	illegal instruction trap
2	BPT/trace trap
3	IOT
4	power fail trap (if autoreboot fails)
5	EMT
6	recursive system call (TRAP instruction)
7	programmed interrupt request
11	protection fault (segmentation violation)
12	parity trap

In some of these cases it is possible for octal 020 to be added into the trap type; this indicates that the processor was in user mode when the trap occurred.

In addition to the trap type, the system will have printed out three (or four) other numbers: ka6, which is the contents of the segmentation register for the area in which the system's stack is kept; aps, which is the location where the hardware stored the program status word during the trap; pc, which was the system's program counter when it faulted (already incremented to the next word); __ovno, the overlay number of the currently loaded kernel overlay when the trap occurred.

The favorite trap types in system crashes are trap types 0 and 11, indicating a wild reference. The code is the referenced address, and the pc at the time of the fault is printed. These problems tend to be easy to track down if they are kernel bugs since the processor stops cold, but random flakiness seems to cause this sometimes. The debugger can be used to locate the instruction and subroutine corresponding to the PC value. If that is insufficient to suggest the nature of the problem, more detailed examination of the system status at the time of the trap usually can produce an explanation.

init died
The system initialization process has exited. This is bad news, as no new users will then be able to log in. Rebooting is the only fix, so the system just does it right away.

out of mbufs: map full
The network has exhausted its private page map for network buffers. This usually indicates that buffers are being lost, and rather than allow the system to slowly degrade, it reboots immediately. The map may be made larger if necessary.

out of swap space
This really shouldn't be panics but there's no other satisfactory solution. The size of the swap area must be increased. The system attempts to avoid running out of swap by refusing to start new processes when short of swap space (resulting in ``No more proceses'' messages from the shell).

&remap_area > SEG5
_end > SEG5
The kernel detected at boot time that an unacceptable portion of its data space extended into the region controlled by KDSA5. In the case of the first message, the size of the kernel's data segment (excluding the file, proc, and text tables) must be decreased. In the latter case, there are two possibilities: if &remap_area is not greater than 0120000, the kernel must be recompiled without defining the option NOKA5. Otherwise, as above, the size of the kernel's data segment must be decreased.

That completes the list of panic types you are likely to see. There are many other panic messages which are less likely to occur; most of them detect logical inconsistencies within the kernel and thus ``cannot happen'' unless some part of the kernel has been modified.

If the system stops or hangs without a panic, it is possible to stop it and take a dump of memory before rebooting. A panic can be forced from the console, which will allow a dump, automatic reboot and file system check. This is accomplished by halting the CPU, putting the processor in kernel mode, loading the PC with 40, and continuing without a reset (use continue, not start). To put the processor in kernel mode, make sure the two high bits in the processor status word are zero. (you'll need to consult the procesor handbook describing your processor to determine how to access the PC and PS ...) The message ``panic: forced from console'' should print, and the automatic reboot will start.

If this fails a dump of memory can be made on magtape: mount a tape (with write ring!), halt the CPU, load address 044, and perform a start (which does a reset). This should write a copy of all of core on the tape with an EOF mark. Caution: Any error is taken to mean the end of core has been reached. This means that you must be sure the ring is in, the tape is ready, and the tape is clean and new. If the dump fails, you can try again, but some of the registers will be lost. After this completes, halt again and reboot.

After rebooting, or after an automatic file system check fails, check and fix the file systems with fsck. If the system will not reboot, a runnable system must be obtained from a backup medium after verifying that the hardware is functioning normally. A damaged root file system should be patched while running with an alternate root if possible.

When the system crashes if crash dumping was enabled it writes (or at least attempts to write) an image of memory into the back end of the dump device, usually the same as the primary swap area. After the system is rebooted, the program savecore(8) runs and preserves a copy of this core image and the current system in a specified directory for later perusal. See savecore(8) for details. A magtape dump can be read onto disk with dd(1).

To analyze a dump you should begin by running adb(1) with the -k flag on the system load image and core dump. If the core image is the result of a panic, the panic message is printed. Normally the command ``$c'' or ``$C'' will provide a stack trace from the point of the crash and this will provide a clue as to what went wrong. ps(1) and pstat(8)canalsobeused to print the process table at the time of the crash via: ps -alxk and pstat -p. If the mapping or the stack frame are incorrect, the following magic locations may be examined in an attempt to find out what went wrong. The registers R0, R1, R2, R3, R4, R5, SP, and KDSA6 (or KISA6 for machines without separate instruction and data) are saved at location 04. If the core dump was taken on disk, these values also appear at 0300. The value of KDSA6 (KISA6) multiplied by 0100 (8) gives the address of the user structure and kernel stack for the running process. Relabel these addresses 0140000 through 0142000. R5 is C's frame or display pointer. Stored at (R5) is the old R5 pointing to the previous stack frame. At (R5)+2 is the saved PC of the calling procedure. Trace this calling chain to an R5 value of 0141756 (0141754 for overlaid kernels), which is where the user's R5 is stored. If the chain is broken, look for a plausible R5, PC pair and continue from there. In most cases this procedure will give an idea of what is wrong.

A more complete discussion of system debugging is impossible here. See, however, ``Using ADB to Debug the UNIX Kernel''.

SEE ALSO

adb(1), ps(1), pstat(1), boot(8), fsck(8), reboot(8), savecore(8)
PDP-11 Processor Handbook for various processors for more information about PDP-11 memory management and general architecture.
Using ADB to Debug the UNIX Kernel

NAME

cron - daemon to execute scheduled commands (Vixie Cron)

SYNOPSIS

cron

DESCRIPTION

Cron should be started from /etc/rc or /etc/rc.local. It will return immediately, so you don't need to start it with '&'.

Cron searches /var/cron/tabs for crontab files which are named after accounts in /etc/passwd; crontabs found are loaded into memory. Cron also searches for /etc/crontab which is in a different format (see crontab(5)). Cron then wakes up every minute, examining all stored crontabs, checking each command to see if it should be run in the current minute. When executing commands, any output is mailed to the owner of the crontab (or to the user named in the MAILTO environment variable in the crontab, if such exists).

Additionally, cron checks each minute to see if its spool directory's modtime (or the modtime on /etc/crontab) has changed, and if it has, cron will then examine the modtime on all crontabs and reload those which have changed. Thus cron need not be restarted whenever a crontab file is modified. Note that the Crontab(1) command updates the modtime of the spool directory whenever it changes a crontab.

SEE ALSO

crontab(1), crontab(5)

AUTHOR

Paul Vixie <paul@vix.com>

NAME

dcheck - file system directory consistency check

SYNOPSIS

dcheck [-i numbers] [filesystem]

DESCRIPTION

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.

SEE ALSO

filsys(5), clri(8), fsck(8), icheck(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.

BUGS

Since dcheck is inherently two-pass in nature, extraneous diagnostics may be produced if applied to active file systems. Default file systems vary with installation so dcheck should use fstab(5).

NAME

diskpart - calculate default disk partition sizes

SYNOPSIS

/usr/sbin/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). On disks that use bad144-style bad-sector forwarding, space is 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 given 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. The device driver tables include the space reserved for the bad sector forwarding table in the `c' partition; those used in the disktab and default formats exclude reserved tracks. 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)

BUGS

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.

NAME

dmesg - collect system diagnostic messages to form error log

SYNOPSIS

/sbin/dmesg [-]

DESCRIPTION

N.B.: Dmesg is obsoleted by syslogd(8) for maintenance of the system error log.

Dmesg looks in a system buffer for recently printed diagnostic messages and prints them on the standard output. The messages are those printed or logged by the system when errors occur. 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.

FILES

/usr/adm/msgbuf scratch file for memory of - option

SEE ALSO

syslogd(8)

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(8V), bad144(8)

AUTHOR

Helge Skrivervik

NAME

dump - incremental file system dump

SYNOPSIS

dump [0123456789BchfusTdWwn [argument ...]] [filesystem]

DESCRIPTION

Dump copies to magnetic tape all files changed after a certain date in the filesystem.

The following options are supported by dump:

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.

B records
The number of dump records per volume. This option overrides the calculation of tape size based on length and density.

C This option requires no further options. Used to specify that the tape is a cartridge drive rather than a 9-track.

h level
Honor the user 'nodump' flags only for dumps at or above the given level. The default honor level is 1, so that incremental backups omit such files but full backups retain them.

F Place the dump on the next argument file instead of the tape. If '-' is given then standard out (stdout) is written to.

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. Note that /etc/dumpdates is in a format different from that which previous versions of dump maintained in /etc/ddate, although the information content is identical.

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.

T date
Use the specified date as the starting time for the dump instead of the time determined from looking in /etc/dumpdates. The format of date is the same as that of ctime(3). This option is useful for automated dump scripts that wish to dump over a specific period of time. The T option is mutually exclusive with the u option.

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/rxp0a	default filesystem to dump from
/dev/rmt0	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

restor(8), rdump(8), dump(5), fstab(5), dumpdir(8)

DIAGNOSTICS

Many, and verbose.

BUGS

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 restor.

NAME

dump, ddate - incremental dump format

SYNOPSIS

```
#include <sys/types.h>
#include <sys/ino.h>
#include <dumprestor.h>
```

DESCRIPTION

Tapes used by dump and restor(8) contain:

a header record
two groups of bit map records
a group of records describing directories
a group of records describing files

The format of the header record and of the first record of each description as given in the include file <dumprestor.h> is:

```
#if  UCB_NKB == 1
#define NTREC    10
#endif
#ifndef          UCB_NKB
#define NTREC    20
#endif
#define MLEN     16
#define MSIZ     4096

#define TS_TAPE  1
#define TS_INODE 2
#define TS_BITS  3
#define TS_ADDR  4
#define TS_END   5
#define TS_CLRI  6
#define MAGIC    (int)60011
#define CHECKSUM (int)84446
struct       spcl
{
   int       c_type;
   time_t    c_date;
   time_t    c_ddate;
   int       c_volume;
   daddr_t   c_tapea;
   ino_t     c_inumber;
   int       c_magic;
   int       c_checksum;
   struct    dinodec_dinode;
   int       c_count;
   char      c_addr[BSIZE];
} spcl;
```

```
struct      idates
{
   char     id_name[16];
   char     id_incno;
   time_t   id_ddate;
};
```

NTREC is the number of BSIZE (sys/param.h) byte records in a physical tape block. MLEN is the number of bits in a bit map word. MSIZ is the number of bit map words.

The TS_ entries are used in the c_type field to indicate what sort of header this is. The types and their meanings are as follows:

TS_TAPE Tape volume label

TS_INODE
A file or directory follows. The c_dinode field is a copy of the disk inode and contains bits telling what sort of file this is.

TS_BITS A bit map follows. This bit map has a one bit for each inode that was dumped.

TS_ADDR A subrecord of a file description. See c_addr below.

TS_END End of tape record.

TS_CLRI A bit map follows. This bit map contains a zero bit for all inodes that were empty on the file system when dumped.

MAGIC All header records have this number in c_magic.

CHECKSUM
Header records checksum to this value.

The fields of the header structure are as follows:

c_type The type of the header.

c_date The date the dump was taken.

c_ddate The date the file system was dumped from.

c_volume The current volume number of the dump.

c_tapea The current number of this (512-byte) record.

c_inumber
The number of the inode being dumped if this is of type TS_INODE.

c_magic This contains the value MAGIC above, truncated as needed.

c_checksum
This contains whatever value is needed to make the record sum to CHECKSUM.

c_dinode This is a copy of the inode as it appears on the file system; see filsys(5).

c_count The count of characters in c_addr.

c_addr An array of characters describing the blocks of the dumped file. A character is zero if the block

associated with that character was not present on the file system, otherwise the character is non-zero. If the block was not present on the file system, no block was dumped; the block will be restored as a hole in the file. If there is not sufficient space in this record to describe all of the blocks in a file, TS_ADDR records will be scattered through the file, each one picking up where the last left off.

Each volume except the last ends with a tapemark (read as an end of file). The last volume ends with a TS_END record and then the tapemark.

The structure idates describes an entry of the file /etc/ddate where dump history is kept. The fields of the structure are:

id_name The dumped file system is `/dev/id_nam'.
id_incno The level number of the dump tape; see dump(8).
id_ddate The date of the incremental dump in system format see types(5).

FILES

/etc/ddate

SEE ALSO

filsys(5), types(5), dump(8), dumpdir(8), restor(8)

NAME

dumpdir, 512dumpdir - print the names of files on a dump tape

SYNOPSIS

```
dumpdir [ f filename ]
512dumpdir [ f filename ]
```

DESCRIPTION

Dumpdir is used to read magtapes dumped with the dump command and list the names and inode numbers of all the files and directories on the tape.

The f option causes filename as the name of the tape instead of the default.

512dumpdir is a version of dumpdir that can read tapes written from 512-byte block file systems.

FILES

/dev/rmt1	default file name
rst*	temporary files

SEE ALSO

dump(8), restor(8)

DIAGNOSTICS

If the dump extends over more than one tape, it may ask you to change tapes. Reply with a newline when the next tape has been mounted.

BUGS

There is redundant information on the tape that could be used in case of tape reading problems. Unfortunately, dumpdir and 512dumpdir don't use it.

NAME

edquota - edit user quotas

SYNOPSIS

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.

BUGS

The format of the temporary file is inscrutable.

NAME

fingerd - remote user information server

SYNOPSIS

fingerd [-s][-l][-p filename]

DESCRIPTION

Fingerd is a simple protocol based on RFC1196 that provides an interface to the Name and Finger programs at several network sites. The program is supposed to return a friendly, human-oriented status report on either the system at the moment or a particular person in depth. There is no required format and the protocol consists mostly of specifying a single ``command line''.

Fingerd listens for TCP requests at port 79. Once connected it reads a single command line terminated by a <CRLF> which is passed to finger(1). Fingerd closes its connections as soon as the output is finished.

If the line is null (i.e. just a <CRLF> is sent) then finger returns a ``default'' report that lists all people logged into the system at that moment.

If a user name is specified (e.g. eric<CRLF>) then the response lists more extended information for only that particular user, whether logged in or not. Allowable ``names'' in the command line include both ``login names'' and ``user names''. If a name is ambiguous, all possible derivations are returned.

The following options may be passed to fingerd as server program arguments in /etc/inetd.conf:

-s Enable secure mode. Queries without a user name are rejected and forwarding of queries to other remote hosts is denied.

-l Enable logging. The name of the host originating the query is reported via syslog(3) at LOG_NOTICE priority.

-p Use an alternate program as the local information provider. The default local program executed by fingerd is finger(1). By specifying a customized local server, this option allows a system manager to have more control over what information is provided to remote sites.

SEE ALSO

finger(1)

BUGS

Connecting directly to the server from a TIP or an equally

narrow-minded TELNET-protocol user program can result in meaningless attempts at option negotiation being sent to the server, which will foul up the command line interpretation. Fingerd should be taught to filter out IAC's and perhaps even respond negatively (IAC WON'T) to all option commands received.

HISTORY

The fingerd command appeared in 4.3BSD.

NAME

fingerd - remote user information server

SYNOPSIS

fingerd [-s][-l][-p filename]

DESCRIPTION

Fingerd is a simple protocol based on RFC1196 that provides an interface to the Name and Finger programs at several network sites. The program is supposed to return a friendly, human-oriented status report on either the system at the moment or a particular person in depth. There is no required format and the protocol consists mostly of specifying a single ``command line''.

Fingerd listens for TCP requests at port 79. Once connected it reads a single command line terminated by a <CRLF> which is passed to finger(1). Fingerd closes its connections as soon as the output is finished.

If the line is null (i.e. just a <CRLF> is sent) then finger returns a ``default'' report that lists all people logged into the system at that moment.

If a user name is specified (e.g. eric<CRLF>) then the response lists more extended information for only that particular user, whether logged in or not. Allowable ``names'' in the command line include both ``login names'' and ``user names''. If a name is ambiguous, all possible derivations are returned.

The following options may be passed to fingerd as server program arguments in /etc/inetd.conf:

-s Enable secure mode. Queries without a user name are rejected and forwarding of queries to other remote hosts is denied.

-l Enable logging. The name of the host originating the query is reported via syslog(3) at LOG_NOTICE priority.

-p Use an alternate program as the local information provider. The default local program executed by fingerd is finger(1). By specifying a customized local server, this option allows a system manager to have more control over what information is provided to remote sites.

SEE ALSO

finger(1)

BUGS

Connecting directly to the server from a TIP or an equally

narrow-minded TELNET-protocol user program can result in meaningless attempts at option negotiation being sent to the server, which will foul up the command line interpretation. Fingerd should be taught to filter out IAC's and perhaps even respond negatively (IAC WON'T) to all option commands received.

HISTORY

The fingerd command appeared in 4.3BSD.

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 too many 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. It is also possible to reformat sections of the disk in units of tracks. 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 burn-in," which makes a number of passes using different patterns. The number of passes can be selected at run time, up to a maximum of 48, with provision for additional passes or termination after the preselected number of passes. This test runs for many hours, depending on the disk and processor.

Each time format is run to format an entire disk, 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 that 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 that all sectors with uncorrectable media errors are remapped. If a drive develops uncorrectable defects after formatting, either bad144(8) or badsect(8) should be able to avoid the bad sectors.

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 with boot(8) following a "B/3" command. 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
Starting cylinder (0):
Starting track (0):
Ending cylinder (841):
```

```
        Ending track (19):
        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 (up to 48 passes)
        Pattern (one of the above, other to restart)? 2
        Maximum number of bit errors to allow for soft ECC (3):
        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:
        Bad sector: 0
        Write check: 4
        Hard ECC: 0
        Other hard: 0
        Marked bad: 0
        Skipped: 0
        Total of 4 hard errors revectored.
        Writing bad sector table at block 808272
        (808272 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)

BUGS

An equivalent facility should be available which operates under a running UNIX system.

It should be possible to reformat or verify part or all of a disk, then update the existing bad sector table.

NAME

fsck - file system consistency check and interactive repair

SYNOPSIS

fsck -p[#] [filesystem ...]
fsck [-y] [-n] [-sX] [-SX] [-t filename] [filesystem] ...

DESCRIPTION

The first form of fsck preens a standard set of file systems 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. Only one file system from each disk should be checked in each pass. 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. A number or range of numbers may be given after the -p to start or end the check at a specified pass number. A single number specifies the starting pass (e.g. -p2 or -p2-); a range specifies the starting and endig passes (-p2-4). A missing start means to start with pass 1 (-p-2). These can be used to stop fsck to mount a temporary file system before continuing the check on larger file systems requiring scratch files.

The system normally 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 superblock wrong

Allocated inodes in the free inode list

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. Warning: kernel changes are required to limit the types of inconsistencies, and fsck -p makes assumptions about the kernel's actions in repairing these. Vmunix, PDP-11 2.9BSD with the UCB_FSFIX option and PDP-11 2.10BSD Unix kernels have the appropriate modifications; the -p option should not be used on other systems.

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.

-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.

-sX Ignore the actual free list and (unconditionally) reconstruct a new one by rewriting the superblock of the file system. The file system should be unmounted while this is done; if this is not possible, care should be taken that the system is quiescent and that it is rebooted immediately afterwards. This precaution is necessary so that the old, bad, in-core copy of the superblock will not continue to be used, or written on the file system.

The -sX option allows for creating an optimal free list organization. The following forms of X are

supported for the following devices:

-s3 (RP03)
-s4 (RP04, RP05, RP06)
-sBlocks-per-cylinder:Blocks-to-skip (for anything else)

If X is not given, the values used when the file system was created are used. If these values were not specified, then the value 400:9 is used.

-SX Conditionally reconstruct the free list. This option is like -sX above except that the free list is rebuilt only if there were no discrepancies discovered in the file system. Using -S will force a no response to all questions asked by fsck. This option is useful for forcing free list reorganization on uncontaminated file systems.

-t If fsck cannot obtain enough memory to keep its tables, it uses a scratch file. If the -t option is specified, the file named in the next argument is used as the scratch file, if needed. Without the -t flag, fsck will prompt the operator for the name of the scratch file. The file chosen should not be on the file system being checked, and if it is not a special file or did not already exist, it is removed when fsck completes.

If no file systems 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 16-byte aligned.
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 than 65536 inodes.
   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.
11. Allocated inodes on the free inode list in the

superblock.

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 file system being checked and must have empty slots in which entries can be made. This can be accomplished manually by making lost+found, copying a number of files to the directory, and then removing them (before fsck is executed). Mkfs(8) will automatically create a lost+found directory.

Checking the raw device is almost always faster. The root device should not be checked using the raw device, however, since it cannot be unmounted.

FILES

/etc/fstab default list of file systems to check

DIAGNOSTICS

The diagnostics produced by fsck are intended to be self-explanatory. The exit codes with the -p option are 0 (no problems that weren't fixed), 4 (root file system was modified), 8 (problems that couldn't be fixed) and 12 (fsck was interrupted).

SEE ALSO

filsys(5), fstab(5), crash(8), mkfs(8), mklost+found(8), reboot(8)
T. J. Kowalski, FSCK - The UNIX File System Check Program

BUGS

Inode numbers for . and .. in each directory should be checked for validity.

NAME

ftpd - DARPA Internet File Transfer Protocol server

SYNOPSIS

/usr/libexec/ftpd [-d] [-l] [-ttimeout] [-Tmaxtimeout]

DESCRIPTION

Ftpd is the DARPA Internet File Transfer Protocol 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, debugging information is written to the syslog.

If the -l option is specified, each ftp session is logged in the syslog.

The ftp server will timeout an inactive session after 15 minutes. If the -t option is specified, the inactivity timeout period will be set to timeout seconds. A client may also request a different timeout period; the maximum period allowed may be set to timeout seconds with the -T option. The default limit is 2 hours.

The ftp server currently supports the following ftp requests; case is not distinguished.

Request	Description
ABOR	abort previous command
ACCT	specify account (ignored)
ALLO	allocate storage (vacuously)
APPE	append to a file
CDUP	change to parent of current working directory
CWD	change working directory
DELE	delete a file
HELP	give help information
LIST	give list files in a directory (``ls -lgA'')
MKD	make a directory
MDTM	show last modification time of file
MODE	specify data transfer mode
NLST	give name list of files in directory
NOOP	do nothing
PASS	specify password
PASV	prepare for server-to-server transfer
PORT	specify data connection port
PWD	print the current working directory
QUIT	terminate session
REST	restart incomplete transfer
RETR	retrieve a file
RMD	remove a directory

RNFR	specify rename-from file name
RNTO	specify rename-to file name
SITE	non-standard commands (see next section)
SIZE	return size of file
STAT	return status of server
STOR	store a file
STOU	store a file with a unique name
STRU	specify data transfer structure
SYST	show operating system type of server system
TYPE	specify data transfer type
USER	specify user name
XCUP	change to parent of current working directory (deprecated)
XCWD	change working directory (deprecated)
XMKD	make a directory (deprecated)
XPWD	print the current working directory (deprecated)
XRMD	remove a directory (deprecated)

The following non-standard or UNIX specific commands are supported by the SITE request.

Request	Description
UMASK	change umask. E.g. SITE UMASK 002
IDLE	set idle-timer. E.g. SITE IDLE 60
CHMOD	change mode of a file. E.g. SITE CHMOD 755 filename
HELP	give help information. E.g. SITE HELP

The remaining ftp requests specified in Internet RFC 959 are recognized, but not implemented. MDTM and SIZE are not specified in RFC 959, but will appear in the next updated FTP RFC.

The ftp server will abort an active file transfer only when the ABOR command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in Internet RFC 959. If a STAT command is received during a data transfer, preceded by a Telnet IP and Synch, transfer status will be returned.

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) The user must have a standard shell returned by getusershell(3).

4) 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 command. 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 be able to produce owner names rather than numbers. The password field in passwd is not used, and should not contain real encrypted passwords. 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(1), getusershell(3), syslogd(8)

BUGS

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.

NAME

gettable - get NIC format host tables from a host

SYNOPSIS

/usr/sbin/gettable [-v] host [outfile]

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 outfile or by default, hosts.txt.

The -v option will get just the version number instead of the complete host table and put the output in the file outfile or by default, hosts.ver.

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), named(8)

BUGS

If the name-domain system provided network name mapping well as host name mapping, gettable would no longer be needed.

NAME

getty - set terminal mode

SYNOPSIS

getty [type [tty]]

DESCRIPTION

Getty is usually invoked by init(8) to open and initialize the tty line, read a login name, and invoke login(1). getty attempts to adapt the system to the speed and type of terminal being used.

The argument tty is the special device file in /dev to open for the terminal (e.g., ``ttyh0''). If there is no argument or the argument is ``-'', the tty line is assumed to be open as file descriptor 0.

The type argument can be used to make getty treat the terminal 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 same 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 an 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.

DIAGNOSTICS

ttyxx: No such device or address. ttyxx: No such file or address. A terminal which is turned on in the ttys file cannot be opened, likely because the requisite lines are either not configured into the system, the associated device was not attached during boot-time system configuration, or the special file in /dev does not exist.

FILES

/etc/gettytab

SEE ALSO

gettytab(5), init(8), login(1), ioctl(2), tty(4), ttys(5)

NAME

reboot - stopping and restarting the system

SYNOPSIS

```
/sbin/reboot   [ -lqnhdarsfRD ]
/sbin/halt     [ -lqndars ]
/sbin/fastboot [ -lqndarsRD ]
```

DESCRIPTION

2.11BSD is started by placing it in memory at location zero and transferring to its entry point. Since the system is not reentrant, it is necessary to read it in from disk or tape each time it is to be boot strapped.

Rebooting a running system: When the system is running and a reboot is desired, shutdown(8) is normally used to stop time sharing and put the system into single user mode. If there are no users then /sbin/reboot can be used without shutting the system down first.

Reboot normally causes the disks to be synced and allows the system to perform other shutdown activities such as resynchronizing hardware time-of-day clocks. A multi-user reboot (as described below) is then 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 multi-user operation.

Options to reboot are:

-l Don't try to tell syslogd(8) what's about to happen.

-q Reboot quickly and ungracefully, without shutting down running processes first.

-n Don't sync before rebooting. This can be used if a disk or the processor is on fire.

-h Don't reboot, simply halt the processor.

-d Dump memory onto the dump device, usually part of swap, before rebooting. The dump is done in the same way as after a panic.

-a Have the system booter ask for the name of the system to be booted, rather than immediately booting the default system (/unix).

-r Mount the root file system as read only when the system reboots. This is not supported by the kernel in 2.11BSD.

-s Don't enter multi-user mode after system has rebooted - stay in single user mode.

-f Fast reboot. Omit the automatic file system consistency check when the system reboots and goes multi-user. This is accomplished by passing a fast reboot flag on to the rebooting kernel. This currently prevents the use of -f flag in conjunction with the -h (halt) flag.

-D Set the autoconfig(8) debug flag. This is normally not used unless one is debugging the autoconfig program.

-R Tells the kernel to use the compiled in root device. Normally the system uses the device from which it was booted as the root/swap/pipe/dump device.

Reboot normally places a shutdown record in the login accounting file /usr/adm/wtmp. This is inhibited if the -q or -n options are present. Note that the -f (fast reboot) and -n (don't sync) options are contradictory; the request for a fast reboot is ignored in this case.

Halt and fastboot are synonymous with ``reboot -h'' and ``reboot -f'', respectively.

Power fail and crash recovery: Normally, the system will reboot itself at power-up or after crashes if the contents of low memory are intact. An automatic consistency check of the file systems will be performed, and unless this fails, the system will resume multi-user operations.

SEE ALSO

autoconfig(8), sync(2), utmp(8), shutdown(8), syslogd(8)

NAME

htable - convert NIC standard format host tables

SYNOPSIS

/etc/htable [-c connected-nets] [-l local-nets] 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 may be used by the gethostbyname(3N) routines in mapping host names to addresses if the nameserver, named(8), is not used. The networks file is used by the getnetent(3N) routines in mapping network names to numbers. The gateways file may be 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. Of these, only the gateways file is interpreted. This allows sites to maintain local aliases and entries which are not normally present in the master database. Only one gateway to each network will be placed in the gateways file; a gateway listed in the localgateways file will override any in the input file.

If the gateways file is to be used, a list of networks to which the host is directly connected is specified with the -c flag. The networks, separated by commas, may be given by name or in Internet-standard dot notation, e.g. -c arpanet,128.32,local-ether-net. Htable only includes gateways which are directly connected to one of the networks specified, or which can be reached from another gateway on a connected net.

If the -l option is given with a list of networks (in the same format as for -c), these networks will be treated as ``local,'' and information about hosts on local networks is taken only from the localhosts file. Entries for local hosts from the main database will be omitted. This allows the localhosts file to completely override any entries in the input file.

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), named(8)

BUGS

If the name-domain system provided network name mapping well

as host name mapping, htable would no longer be needed.

NAME

icheck - file system storage consistency check

SYNOPSIS

icheck [-s] [-b numbers] [filesystem]

DESCRIPTION

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, quota nodes, and symbolic links.

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.

SEE ALSO

filsys(5), clri(8), dcheck(8), fsck(8), ncheck(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.

BUGS

Since icheck is inherently two-pass in nature, extraneous diagnostics may be produced if applied to active file systems. Since default file systems vary with installations, icheck should use fstab(5).
It believes even preposterous super-blocks and consequently can get core images.

NAME

ifconfig - configure network interface parameters

SYOPNSIS

/sbin/ifconfig interface address_family [address [dest_address]] [parameters]
/sbin/ifconfig interface [protocol_family]

DESCRIPTION

Ifconfig is used to assign an address to a network interface and/or configure network interface parameters. Ifconfig 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 or other operating parameters. The interface parameter is a string of the form ``name unit'', e.g. ``en0''.

Since an interface may receive transmissions in differing protocols, each of which may require separate naming schemes, it is necessary to specify the address_family, which may change the interpretation of the remaining parameters. The address families currently supported are ``inet'' and ``ns''.

For the DARPA-Internet family, 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''. For the Xerox Network Systems(tm) family, addresses are net:a.b.c.d.e.f, where net is the assigned network number (in decimal), and each of the six bytes of the host number, a through f, are specified in hexadecimal. The host number may be omitted on 10Mb/s Ethernet interfaces, which use the hardware physical address, and on interfaces other than the first.

The following parameters may be set with ifconfig:

up Mark an interface ``up''. This may be used to enable an interface after an ``ifconfig down.'' It happens automatically when setting the first address on an interface. If the interface was reset when previously marked down, the hardware will be re-initialized.

down Mark an interface ``down''. When an interface is marked ``down'', the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.

trailers Request the use of a ``trailer'' link level encapsulation when sending (default). If a network interface supports trailers, the system will, when possible, encapsulate outgoing messages in a manner which minimizes the number of memory to memory copy operations performed by the receiver. On networks that support the Address Resolution Protocol (see arp(4P); currently, only 10 Mb/s Ethernet), this flag indicates that the system should request that other systems use trailers when sending to this host. Similarly, trailer encapsulations will be sent to other hosts that have made such requests. Currently used by Internet protocols only.

-trailers Disable the use of a ``trailer'' link level encapsulation.

arp Enable the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between DARPA Internet addresses and 10Mb/s Ethernet addresses.

-arp Disable the use of the Address Resolution Protocol.

metric n Set the routing metric of the interface to n, default 0. The routing metric is used by the routing protocol (routed(8)). Higher metrics have the effect of making a route less favorable; metrics are counted as addition hops to the destination network or host.

debug Enable driver dependent debugging code; usually, this turns on extra console error logging.

-debug Disable driver dependent debugging code.

netmask mask (Inet only) Specify how much of the address to reserve for subdividing networks into sub-networks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number with a leading 0x, with a dot-notation Internet address, or with a pseudo-network name listed in the network table networks(5). The mask contains 1's for

the bit positions in the 32-bit address which are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion.

dstaddr Specify the address of the correspondent on the other end of a point to point link.

broadcast (Inet only) Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 1's.

ipdst (NS only) This is used to specify an Internet host who is willing to receive ip packets encapsulating NS packets bound for a remote network. In this case, an apparent point to point link is constructed, and the address specified will be taken as the NS address and network of the destinee.

Ifconfig displays the current configuration for a network interface when no optional parameters are supplied. If a protocol family is specified, Ifconfig will report only the details specific to that protocol family.

Only the super-user may modify the configuration of a network interface.

DIAGNOSTICS

Messages indicating the specified interface does not exit, the requested address is unknown, or the user is not privileged and tried to alter an interface's configuration.

SEE ALSO

netstat(1), intro(4N), rc(8)

NAME

implog - IMP log interpreter

SYNOPSIS

/usr/sbin/implog [-D] [-f] [-c] [-r] [-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.

-r Print the raw imp leader, showing all fields, in addition to the formatted interpretation according to type.

-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)

BUGS

Can not specify multiple hosts, imps, etc. Can not follow reception of messages without looking at those currently in

the file.

NAME

implogd - IMP logger process

SYNOPSIS

/usr/sbin/implogd [-d]

DESCRIPTION

Implogd is program which logs error 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, ``host down'' and other error messages, and, perhaps, stray NCP messages.

SEE ALSO

imp(4P), implog(8C)

NAME

inetd - internet ``super-server''

SYNOPSIS

inetd [-d] [-R rate] [configuration file]

DESCRIPTION

The inetd program should be run at boot time by /etc/rc (see rc(8)). It then listens for connections on certain internet sockets. When a connection is found on one of its sockets, it decides what service the socket corresponds to, and invokes a program to service the request. The server program is invoked with the service socket as its standard input, output and error descriptors. After the program is finished, inetd continues to listen on the socket (except in some cases which will be described below). Essentially, inetd allows running one daemon to invoke several others, reducing load on the system.

The options available for inetd:

-d Turns on debugging.

-R rate Specifies the maximum number of times a service can be invoked in one minute; the default is 1000.

Upon execution, inetd reads its configuration information from a configuration file which, by default, is /etc/inetd.conf. There must be an entry for each field of the configuration file, with entries for each field separated by a tab or a space. Comments are denoted by a ``#'' at the beginning of a line. There must be an entry for each field. The fields of the configuration file are as follows:

service name
socket type
protocol
wait/nowait
user
server program
server program arguments

There are two types of services that inetd can start: standard and TCPMUX. A standard service has a well-known port assigned to it; it may be a service that implements an official Internet standard or is a BSD-specific service. As described in RFC 1078, TCPMUX services are nonstandard services that do not have a well-known port assigned to them. They are invoked from inetd when a program connects to the ``tcpmux'' well-known port and specifies the service name. This feature is useful for adding locally-developed servers.

The service-name entry is the name of a valid service in the file /etc/services. For ``internal'' services (discussed below), the service name must be the official name of the service (that is, the first entry in /etc/services). For TCPMUX services, the value of the service-name field consists of the string ``tcpmux'' followed by a slash and the locally-chosen service name. The service names listed in /etc/services and the name ``help'' are reserved. Try to choose unique names for your TCPMUX services by prefixing them with your organization's name and suffixing them with a version number.

The socket-type should be one of ``stream'', ``dgram'', ``raw'', ``rdm'', or ``seqpacket'', depending on whether the socket is a stream, datagram, raw, reliably delivered message, or sequenced packet socket. TCPMUX services must use ``stream''.

NOTE: ``rdm'' and ``seqpacket'' are not supported in 2.11BSD.

The protocol must be a valid protocol as given in /etc/protocols. Examples might be ``tcp'' or ``udp''. TCPMUX services must use ``tcp''.

The wait/nowait entry specifies whether the server that is invoked by inetd will take over the socket associated with the service access point, and thus whether inetd should wait for the server to exit before listening for new service requests. Datagram servers must use ``wait'', as they are always invoked with the original datagram socket bound to the specified service address. These servers must read at least one datagram from the socket before exiting. If a datagram server connects to its peer, freeing the socket so inetd can received further messages on the socket, it is said to be a ``multi-threaded'' server; it should read one datagram from the socket and create a new socket connected to the peer. It should fork, and the parent should then exit to allow inetd to check for new service requests to spawn new servers. Datagram servers which process all incoming datagrams on a socket and eventually time out are said to be ``single-threaded''. Comsat(8), biff(1) and talkd(8) are examples of the latter type of datagram server. Tftpd(8) is an example of a multi-threaded datagram server.

Servers using stream sockets generally are multi-threaded and use the ``nowait'' entry. Connection requests for these services are accepted by inetd , and the server is given only the newly-accepted socket connected to a client of the service. Most stream-based services operate in this manner. Stream-based servers that use ``wait'' are started with the listening service socket, and must accept at least one

connection request before exiting. Such a server would normally accept and process incoming connection requests until a timeout. TCPMUX services must use ``nowait''.

The user entry should contain the user name of the user as whom the server should run. This allows for servers to be given less permission than root.

The server-program entry should contain the pathname of the program which is to be executed by inetd when a request is found on its socket. If inetd provides this service internally, this entry should be ``internal''.

The server program arguments should be just as arguments normally are, starting with argv[0], which is the name of the program. If the service is provided internally, the word ``internal'' should take the place of this entry.

The inetd program provides several ``trivial'' services internally by use of routines within itself. These services are ``echo'', ``discard'', ``chargen'' (character generator), ``daytime'' (human readable time), and ``time'' (machine readable time, in the form of the number of seconds since midnight, January 1, 1900). All of these services are tcp based. For details of these services, consult the appropriate RFC from the Network Information Center.

The inetd program rereads its configuration file when it receives a hangup signal, SIGHUP. Services may be added, deleted or modified when the configuration file is reread.

TCPMUX

RFC 1078 describes the TCPMUX protocol: ``A TCP client connects to a foreign host on TCP port 1. It sends the service name followed by a carriage-return line-feed <CRLF>. The service name is never case sensitive. The server replies with a single character indicating positive (+) or negative (-) acknowledgment, immediately followed by an optional message of explanation, terminated with a <CRLF>. If the reply was positive, the selected protocol begins; otherwise the connection is closed.'' The program is passed the TCP connection as file descriptors 0 and 1.

If the TCPMUX service name begins with a ``+'', inetd returns the positive reply for the program. This allows you to invoke programs that use stdin/stdout without putting any special server code in them.

The special service name ``help'' causes inetd to list TCPMUX services in inetd.conf.

EXAMPLES

Here are several example service entries for the various types of services:

```
ftp               stream tcp nowait root /usr/libexec/ftpd ftpd -l
ntalk             dgram udp wait root /usr/libexec/ntalkd ntalkd
tcpmux/+date      stream tcp nowait guest /bin/date date
tcpmux/phonebook  stream tcp nowait guest /usr/local/phonebook phonebook
```

ERROR MESSAGES

The inetd server logs error messages using syslog(3). Important error messages and their explanations are:

service/protocol server failing (looping), service terminated.
The number of requests for the specified service in the past minute exceeded the limit. The limit exists to prevent a broken program or a malicious user from swamping the system. This message may occur for several reasons: 1) there are lots of hosts requesting the service within a short time period, 2) a 'broken' client program is requesting the service too frequently, 3) a malicious user is running a program to invoke the service in a 'denial of service' attack, or 4) the invoked service program has an error that causes clients to retry quickly. Use the -R option, as described above, to change the rate limit. Once the limit is reached, the service will be reenabled automatically in 10 minutes.

service/protocol: No such user 'user', service ignored
service/protocol: getpwnam: user: No such user
No entry for user exists in the passwd file. The first message occurs when inetd (re)reads the configuration file. The second message occurs when the service is invoked.

service: can't set uid number
service: can't set gid number
The user or group ID for the entry's user is invalid.

SEE ALSO

comsat(8), fingerd(8), ftpd(8), rexecd(8), rlogind(8), rshd(8), telnetd(8), tftpd(8)

HISTORY

The inetd command appeared in 4.3BSD. TCPMUX is based on code and documentation by Mark Lottor.

NAME

init - process control initialization

SYNOPSIS

/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 executes a command for each terminal specified in the file. This command will usually be /usr/libexec/getty. Getty opens and initializes the terminal line, 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 the wtmp, file 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 terminal 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

/usr/libexec/getty gettyargs failing, sleeping. A process being started to service a line is exiting quickly each time it is started. This is often caused by a ringing or noisy terminal line. Init will sleep for 30 seconds,

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*, /var/run/utmp, /usr/adm/wtmp, /etc/ttys, /etc/rc

SEE ALSO

login(1), kill(1), sh(1), ttys(5), crash(8), getty(8), rc(8), reboot(8), halt(8), shutdown(8)

NAME

kgmon - generate a dump of the operating system's profile buffers

SYNOPSIS

/usr/sbin/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.

NAME

lpc - line printer control program

SYNOPSIS

/usr/sbin/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 any temporary files, data files, and control files that cannot be printed (i.e., do not form a complete printer job) from the specified printer queue(s) on the local machine.

disable { all | printer ... }
Turn the specified printer queues off. This prevents new printer jobs from being entered into the queue by lpr.

down { all | printer } message ...
Turn the specified printer queue off, disable printing and put message in the printer status file. The message doesn't need to be quoted, the remaining arguments are treated like echo(1). This is normally used to take a printer down and let others know why (lpq will indicate the printer is down and print the status message).

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.

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. If the user is the super-user, try to abort the current daemon first (i.e., kill and restart a stuck daemon).

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.

up { all | printer ... }
Enable everything and start a new printer daemon. Undoes the effects of down.

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

NAME

lpd - line printer daemon

SYNOPSIS

/usr/sbin/lpd [-l] [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 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 or /etc/hosts.lpd. 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 minfree in each spool directory contains the number of disk blocks to leave free so that the line printer queue won't completely fill the disk. The minfree file can be edited with your favorite text editor.

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).

n Ditroff File. The file contains device independent troff output.

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 cif-plot.

v The file contains a raster image.

r The file contains text data with FORTRAN carriage con-trol 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 logged via syslog(3) using the LOG_LPR facility. 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
/usr/spool/*/minfree	minimum free space to leave
/dev/lp*	line printer devices
/dev/printer	socket for local requests
/etc/hosts.equiv	lists machine names allowed printer access
/etc/hosts.lpd	lists machine names allowed printer access, but not under same administrative control.

SEE ALSO

lpc(8), pac(1), lpr(1), lpq(1), lprm(1), syslog(3), printcap(5)
4.2BSD Line Printer Spooler Manual

NAME

makedev - make system special files

SYNOPSIS

/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 super-user.

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)

NAME

makekey - generate encryption key

SYNOPSIS

/usr/sbin/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)

NAME

mkfs - construct a file system

SYNOPSIS

/sbin/mkfs [-i bytes] [-s size] [-m gap] [-n modulus] special

DESCRIPTION

N.B.: file systems are normally created with the newfs(8) command.

Mkfs constructs a file system by writing on the special file special. The size of the filesystem in logical blocks is specified by the -s size option. Logical blocks are 1K (2 sectors) under 2.11BSD.

> NOTE: The newfs(8) program's -s option is in units of sectors. Newfs(8) converts this to filesystem (logical) blocks for mkfs(8).

The number of inodes is calculated based on the argument bytes to the -i option. The default is 4096. If more inodes are desired in a filesystem (there is an absolute maximum of 65500) then a lower value for bytes should be used, perhaps 3072 or even 2048.

The flags -m gap and -n modulus determine the block interleaving of the freelist that will be constructed, where gap is the distance between successive 1024-byte blocks, and modulus is the number of blocks before the pattern repeats, typically one cylinder. The optimal values for these parameters vary with the speed and geometry of the disk, as well as the speed of the processor. Newfs(8) will calculate the correct values in almost all cases from the disklabel.

SEE ALSO

fs(5), dir(5), disklabel(8), fsck(8), mkproto(8) newfs(8)

BUGS

The lost+found directory is created but the boot block is left uninitialized (see disklabel(8).)

NAME

mkhosts - generate hashed host table

SYNOPSIS

/usr/sbin/mkhosts [-v] hostfile

DESCRIPTION

Mkhosts is used to generated the hashed host database used by one version of the library routines gethostbyaddr() and gethostbyname(). It is not used if host name translation is performed by named(8). If the -v option is supplied, each host will be listed as it is added. The file hostfile is usually /etc/hosts, and in any case must be in the format of /etc/hosts (see hosts(5)). Mkhosts will generate database files named hostfile.pag and hostfile.dir. The new database is build in a set of temporary files and only replaces the real database if the new one is built without errors. Mkhosts will exit with a non-zero exit code if any errors are detected.

FILES

hostfile.pag - real database filenames
hostfile.dir
hostfile.new.pag - temporary database filenames
hostfile.new.dir

SEE ALSO

gethostbyname(3), gettable(8), hosts(5), htable(8), named(8)

NAME

mklost+found - make a lost+found directory for fsck

SYNOPSIS

/usr/sbin/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 is obsolete because fsck(8) automatically creates and extends the lost+found directory as needed.

SEE ALSO

fsck(8), mkfs(8)

NAME

mknod - build special file

SYNOPSIS

/sbin/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), makedev(8)

NAME

mkpasswd - generate hashed password table

SYNOPSIS

mkpasswd [-p] passwdfile

DESCRIPTION

Mkpasswd is used to generated the hashed password database used by the password library routines (see getpwent(3)).

The file passwdfile must be in password file format (see passwd(5)). Mkpasswd generates database files named ``passwdfile.pag'' and ``passwdfile.dir'' (see ndbm(3)). Data subsequently taken from the database files differ from passwdfile in one respect. Rather than storing the encrypted password in the database, mkpasswd stores the offset of the encrypted password in passwdfile.

Mkpasswd exits zero on success, non-zero on failure.

The -p option causes mkpasswd to create ``passwdfile.orig'', a password file in the standard Version 7 format.

SEE ALSO

chpass(1), passwd(1), getpwent(3), ndbm(3), passwd(5), vipw(8)

NAME

mkproto - construct a prototype file system

SYNOPSIS

/usr/sbin/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)

BUGS

There should be some way to specify links.

There should be some way to specify bad blocks.

Mkproto can only be run on virgin file systems. It should be possible to copy files into existent file systems.

Mkproto can only copy files up to a single level indirect less 4kb. This works out to about 252Kb

NAME

mount - mount file systems

SYNOPSIS

```
mount [ -adfruvw ] [ -t ufs | external_type ]
mount [ -dfruvw ] special | node
mount [ -dfruvw ] [ -o options ] [ -t ufs | external_type ]
special node
```

DESCRIPTION

The mount command calls the mount(2) system call to prepare and graft a special device on to the file system tree at the point node. If either special or node are not provided, the appropriate information is taken from the fstab(5) file.

The system maintains a list of currently mounted file systems. If no arguments are given to mount, this list is printed.

The options are as follows:

-a Causes everything to be done except for the actual system call. This option is useful in conjunction with the

-v flag to determine what the mount command is trying to do.

-f Forces the revocation of write access when trying to downgrade a filesystem mount status from read-write to read-only. For 2.11BSD this flag is currently not implemented.

-o Options are specified with a -o flag followed by a comma separated string of options. The following options are available:

async All I/O to the file system should be done asynchronously.

This is a dangerous flag to and should not be used unless system should your system crash.

force The same as -f; forces the revocation of write access when trying to downgrade a filesystem mount status from read-write to read-only. This is not (and likely never will be) supported in 2.11BSD.

noaccesstime
File access times are not updated.

This is a performance optimization for read-only, short-lived data, e.g., news.

noauto This filesystem should be skipped when mount is run with the -a flag.

na Same as noauto.

nodev Do not interpret character or block special devices on the file system. This option is useful for a server that has file systems containing special devices for architectures other than its own.

noexec Do not allow execution of any binaries on the mounted file system. This option is useful for a server that has file systems containing binaries for architectures other than its own.

nosuid Do not allow set-user-identifier or set-group-identifier bits to take effect.

rdonly The same as -r; mount the file system read-only (even the super-user may not write it).

sync All I/O to the file system should be done synchronously.

update The same as -u; indicate that the status of an already mounted file system should be changed.

Any additional options specific to a filesystem type that is not one of the internally known types (see the -t option) may be passed as a comma separated list; these options are distinguished by a leading - (dash). Options that take a value are specified using the syntax -option=value. At present no 2.11BSD mount options use the following form, the example has been retained for illustrative purposes only. For example, the mount command:

mount -t mfs -o nosuid,-N,-s=4000 /dev/dk0b /tmp

causes mount to execute the equivalent of:

```
/sbin/mount_mfs -o nosuid -N -s 4000 /dev/dk0b
/tmp
```

-r The file system is to be mounted read-only. Mount the file system read-only (even the super-user may not write it). The same as the ``rdonly'' argument to the -o option.

-t "ufs | external type"
The argument following the -t is used to indicate the file system type. The type ufs is the default. Ufs is also the only value supported by 2.11BSD other than swap. Thus the -t will rarely be used. The -t option can be used to indicate that the actions should only be taken on filesystems of the specified type. More than one type may be specified in a comma separated list. The list of filesystem types can be prefixed with ``no'' to specify the filesystem types for which action should not be taken. For example, the mount command:

```
mount -a -t nonfs,mfs
```

mounts all filesystems except those of type NFS and MFS.

If the type is not one of the internally known types, mount will attempt to execute a program in /sbin/mount_XXX where XXX is replaced by the type name. For example, mfs filesystems are mounted by the program /sbin/mount_mfs.

-u The -u flag indicates that the status of an already mounted file system should be changed. Any of the options discussed above (the -o option) may be changed; also a file system can be changed from read-only to read-write or vice versa. An attempt to change from read-write to read-only will fail if any files on the filesystem are currently open for writing unless the -f flag is also specified. This is currently not implemented in 2.11BSD. The ability to change the flags (nodev, nosuid, etc) is however supported. The set of options is determined by first extracting the options for the file system from the fstab table, then applying any options specified by the -o argument, and finally applying the -r or -w option.

-v Verbose mode.

-w The file system object is to be read and write.

FILES

/etc/fstab
file system table

SEE ALSO

mount(2), fstab(5), umount(8)

BUGS

It is possible for a corrupted file system to cause a crash.

mount and this manpage were ported from 4.4BSD-Lite to 2.11BSD to gain the ability to set the various flags such as nodev, nosuid and so on. Multiple filesystem types are not supported and several of the options and flags are not implemented.

HISTORY

A mount command appeared in Version 6 AT&T UNIX.

NAME

named - Internet domain name server

SYNOPSIS

named [-d debuglevel] [-p port#] [{-b} bootfile]

DESCRIPTION

Named is the Internet domain name server. See RFC883 for more information on the Internet name-domain system. Without any arguments, named will read the default boot file /etc/named.boot, read any initial data and listen for queries.

Options are:

-d Print debugging information. A number after the ``d'' determines the level of messages printed.

-p Use a different port number. The default is the standard port number as listed in /etc/services.

-b Use an alternate boot file. This is optional and allows you to specify a file with a leading dash.

Any additional argument is taken as the name of the boot file. The boot file contains information about where the name server is to get its initial data. If multiple boot files are specified, only the last is used. Lines in the boot file cannot be continued on subsequent lines. The following is a small example:

```
;
;    boot file for name server
;
directory /usr/local/domain

; type    domain                 source host/file         backup file

cache      .                                              root.cache
primary    Berkeley.EDU           berkeley.edu.zone
primary    32.128.IN-ADDR.ARPA    ucbhosts.rev
secondary  CC.Berkeley.EDU        128.32.137.8 128.32.137.3 cc.zone.bak
secondary  6.32.128.IN-ADDR.ARPA  128.32.137.8 128.32.137.3 cc.rev.bak
primary    0.0.127.IN-ADDR.ARPA                           localhost.rev
forwarders 10.0.0.78 10.2.0.78
; slave
```

The ``directory'' line causes the server to change its working directory to the directory specified. This can be important for the correct processing of $INCLUDE files in primary zone files.

The ``cache'' line specifies that data in ``root.cache'' is to be placed in the backup cache. Its main use is to specify data such as locations of root domain servers. This cache is not used during normal operation, but is used as ``hints'' to find the current root servers. The file ``root.cache'' is in the same format as ``berkeley.edu.zone''. There can be more than one ``cache'' file specified. The cache files are processed in such a way as to preserve the time-to-live's of data dumped out. Data for the root nameservers is kept artificially valid if necessary.

The first ``primary'' line states that the file ``berkeley.edu.zone'' contains authoritative data for the ``Berkeley.EDU'' zone. The file ``berkeley.edu.zone'' contains data in the master file format described in RFC883. All domain names are relative to the origin, in this case, ``Berkeley.EDU'' (see below for a more detailed description). The second ``primary'' line states that the file ``ucbhosts.rev'' contains authoritative data for the domain ``32.128.IN-ADDR.ARPA,'' which is used to translate addresses in network 128.32 to hostnames. Each master file should begin with an SOA record for the zone (see below).

The first ``secondary'' line specifies that all authoritative data under ``CC.Berkeley.EDU'' is to be transferred from the name server at 128.32.137.8. If the transfer fails it will try 128.32.137.3 and continue trying the addresses, up to 10, listed on this line. The secondary copy is also authoritative for the specified domain. The first non-dotted-quad address on this line will be taken as a filename in which to backup the transfered zone. The name server will load the zone from this backup file if it exists when it boots, providing a complete copy even if the master servers are unreachable. Whenever a new copy of the domain is received by automatic zone transfer from one of the master servers, this file will be updated. The second ``secondary'' line states that the address-to-hostname mapping for the subnet 128.32.136 should be obtained from the same list of master servers as the previous zone.

The ``forwarders'' line specifies the addresses of sitewide servers that will accept recursive queries from other servers. If the boot file specifies one or more forwarders, then the server will send all queries for data not in the cache to the forwarders first. Each forwarder will be asked in turn until an answer is returned or the list is exhausted. If no answer is forthcoming from a forwarder, the server will continue as it would have without the forwarders line unless it is in ``slave'' mode. The forwarding facility is useful to cause a large sitewide cache to be generated on a master, and to reduce traffic over links to

outside servers. It can also be used to allow servers to run that do not have access directly to the Internet, but wish to act as though they do.

The ``slave'' line (shown commented out) is used to put the server in slave mode. In this mode, the server will only make queries to forwarders. This option is normally used on machine that wish to run a server but for physical or administrative reasons cannot be given access to the Internet, but have access to a host that does have access.

The ``sortlist'' line can be used to indicate networks that are to be preferred over other, unlisted networks. Queries for host addresses from hosts on the same network as the server will receive responses with local network addresses listed first, then addresses on the sort list, then other addresses. This line is only acted on at initial startup. When reloading the nameserver with a SIGHUP, this line will be ignored.

The master file consists of control information and a list of resource records for objects in the zone of the forms:

```
$INCLUDE <filename> <opt_domain>
$ORIGIN <domain>
<domain> <opt_ttl> <opt_class> <type> <resource_record_data>
```

where domain is "." for root, "@" for the current origin, or a standard domain name. If domain is a standard domain name that does not end with ``.'', the current origin is appended to the domain. Domain names ending with ``.'' are unmodified. The opt_domain field is used to define an origin for the data in an included file. It is equivalent to placing a $ORIGIN statement before the first line of the included file. The field is optional. Neither the opt_domain field nor $ORIGIN statements in the included file modify the current origin for this file. The opt_ttl field is an optional integer number for the time-to-live field. It defaults to zero, meaning the minimum value specified in the SOA record for the zone. The opt_class field is the object address type; currently only one type is supported, IN, for objects connected to the DARPA Internet. The type field contains one of the following tokens; the data expected in the resource_record_data field is in parentheses.

A a host address (dotted quad)

NS an authoritative name server (domain)

MX a mail exchanger (domain)

CNAME the canonical name for an alias (domain)

SOA marks the start of a zone of authority (domain of originating host, domain address of maintainer, a serial number and the following parameters in seconds: refresh, retry, expire and minimum TTL (see RFC883))

MB a mailbox domain name (domain)

MG a mail group member (domain)

MR a mail rename domain name (domain)

NULL a null resource record (no format or data)

WKS a well know service description (not implemented yet)

PTR a domain name pointer (domain)

HINFO host information (cpu_type OS_type)

MINFO mailbox or mail list information (request_domain error_domain)

Resource records normally end at the end of a line, but may be continued across lines between opening and closing parentheses. Comments are introduced by semicolons and continue to the end of the line.

Each master zone file should begin with an SOA record for the zone. An example SOA record is as follows:

```
@       IN   SOA  ucbvax.Berkeley.EDU. rwh.ucbvax.Berkeley.EDU. (
             2.89 ; serial
             10800        ; refresh
             3600 ; retry
             3600000   ; expire
             86400 )   ; minimum
```

The SOA lists a serial number, which should be changed each time the master file is changed. Secondary servers check the serial number at intervals specified by the refresh time in seconds; if the serial number changes, a zone transfer will be done to load the new data. If a master server cannot be contacted when a refresh is due, the retry time specifies the interval at which refreshes should be attempted until successful. If a master server cannot be contacted within the interval given by the expire time, all data from the zone is discarded by secondary servers. The minimum value is the time-to-live used by records in the file with no explicit time-to-live value.

NOTES

The boot file directives ``domain'' and ``suffixes'' have been obsoleted by a more useful resolver based implementation of suffixing for partially qualified domain names. The prior mechanisms could fail under a number of situations, especially when then local nameserver did not have complete information.

The following signals have the specified effect when sent to the server process using the kill(1) command.

SIGHUP
Causes server to read named.boot and reload database.

SIGINT
Dumps current data base and cache to /usr/tmp/named_dump.db

SIGIOT
Dumps statistics data into /usr/tmp/named.stats if the server is compiled -DSTATS. Statistics data is appended to the file.

SIGSYS
Dumps the profiling data in /usr/tmp if the server is compiled with profiling (server forks, chdirs and exits).

SIGTERM
Dumps the primary and secondary database files. Used to save modified data on shutdown if the server is compiled with dynamic updating enabled.

SIGUSR1
Turns on debugging; each SIGUSR1 increments debug level. (SIGEMT on older systems without SIGUSR1)

SIGUSR2
Turns off debugging completely. (SIGFPE on older systems without SIGUSR2)

FILES

/etc/named.boot	name server configuration boot file
/var/run/named.pid	the process id
/usr/tmp/named.run	debug output
/usr/tmp/named_dump.db	dump of the name server database
/usr/tmp/named.stats	nameserver statistics data

SEE ALSO

kill(1), gethostbyname(3), signal(3), resolver(3), resolver(5), hostname(7), RFC882, RFC883, RFC973, RFC974, Name Server Operations Guide for BIND

NAME

ncheck - generate names from i-numbers

SYNOPSIS

ncheck [-i numbers] [-a] [-s] [filesystem]

DESCRIPTION

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 file system structure is improper, `??' denotes the `parent' of a parentless file and a pathname beginning with `...' denotes a loop.

NAME

newfs - construct a new file system

SYNOPSIS

/sbin/newfs [-N] [-m free-gap] [-n free-modulus] [-i bytes] [-s size] [-T disk-type] special

DESCRIPTION

Newfs is a ``friendly'' front-end to the mkfs(8) program. Newfs(8) will normally read the disklabel from the drive to determine the partition sizes. If the driver for the disk does not support disklabels the -T option must be used to force a search of /etc/disktab for partition information about drive-type. Newfs calculates the appropriate parameters to use in calling mkfs, then builds the file system by forking mkfs.

-N causes the mkfs command which would be executed to be printed out without actually creating the file system. The disk specified by special must be online though so that newfs can read the disklabel.

-m allows the specification of the block interleaving of the free list. If not specified or outside the range 1 thru 32 then a value of 2 is used.

-n parameter is the freelist modulus (when the -m pattern repeats) and is calculated by newfs to be 1 cylinder in size by default.

-i specifies how many bytes per inode to assume when calculating how many inodes to allocate. The default is 4096 bytes per inode. If this results in too few inodes being allocated (there is an absolute maximum of 65500) then decrease the bytes number (which must lie between 512 and 65536).

-T must be used if the disk specified by special has not been labeled with the disklabel(8) program. In this case disk-type is used by getdisklabel(3) when searching /etc/disktab. This option is used when the underlying device driver does not support disklabels. Care must be taken that the contents of /etc/disktab match the partition tables in the kernel.

-s specifies how many sectors the file system is to contain. There are two sectors per file system block, therefore size should be even. This parameter must be less than or equal to the partition size (as determined from the disklabel or /etc/disktab). An error is printed and no action is taken if the partition size is 0 or too large.

NOTE: Mkfs deals in units of filesystem blocks not sectors. Newfs uses sectors.

FILES

/etc/disktab disk geometry and partition information
mkfs to actually build the file system

SEE ALSO

getdisklabel(3), disklabel(8), disktab(5), diskpart(8), fs(5), fsck(8), mkfs(8)

BUGS

newfs(8) no longer places boot blocks on the filesystem. That duty has been moved to the disklabel(8) program. If you must place a boot block on a disk whose driver does not support disklabels use dd(1).

NAME

pac - printer/plotter accounting information

SYNOPSIS

/usr/sbin/pac [-Pprinter] [-pprice] [-s] [-r] [-c] [-m] [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 or the price specified in /etc/printcap.

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.

The -m flag causes the host name to be ignored in the accounting file. This allows for a user on multiple machines to have all of his printing charges grouped together.

FILES

/usr/adm/?acct	raw accounting files
/usr/adm/?_sum	summary accounting files
/etc/printcap	printer capability data base

SEE ALSO

printcap(5)

BUGS

The relationship between the computed price and reality is as yet unknown.

NAME

ping - send ICMP ECHO_REQUEST packets to network hosts

SYNOPSIS

ping [-dfnqrvR] [-c count] [-i wait] [-l preload] [-p pattern] [-s packetsize]

DESCRIPTION

Ping uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams (``pings'') have an IP and ICMP header, followed by a ``struct timeval'' and then an arbitrary number of ``pad'' bytes used to fill out the packet. The options are as follows:

-c count Stop after sending (and receiving) count ECHO_RESPONSE packets.

-d Set the SO_DEBUG option on the socket being used.

-f Flood ping. Outputs packets as fast as they come back or one hundred times per second, whichever is more. For every ECHO_REQUEST sent a period ``.'' is printed, while for every ECHO_REPLY received a backspace is printed. This provides a rapid display of how many packets are being dropped. Only the super-user may use this option. This can be very hard on

-i wait Wait wait seconds between sending each packet. The default is to wait for one second between each packet. This option is incompatible with the -f option.

-l preload If preload is specified, ping sends that many packets as fast as possible before falling into its normal mode of behavior.

-n Numeric output only. No attempt will be made to lookup symbolic names for host addresses.

-p pattern You may specify up to 16 ``pad'' bytes to fill out the packet you send. This is useful for diagnosing data-dependent problems in a network. For example, `` -p ff'' will cause the sent packet to be filled with all ones.

-q Quiet output. Nothing is displayed except the summary lines at startup time and when finished.

-R Record route. Includes the RECORD_ROUTE option in the ECHO_REQUEST packet and displays the route buffer on returned packets. Note that the IP header is only large enough for nine such routes. Many hosts ignore or discard this option.

-r Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it (e.g., after the interface was dropped by routed(8).

-s packetsize Specifies the number of data bytes to be sent. The default is 56, which translates into 64 ICMP data bytes when combined with the 8 bytes of ICMP header data.

-v Verbose output. ICMP packets other than ECHO_RESPONSE that are received are listed.

When using ping for fault isolation, it should first be run on the local host, to verify that the local network interface is up and running. Then, hosts and gateways further and further away should be ``pinged''. Round-trip times and packet loss statistics are computed. If duplicate packets are received, they are not included in the packet loss calculation, although the round trip time of these packets is used in calculating the minimum/average/maximum round-trip time numbers. When the specified number of packets have been sent (and received) or if the program is terminated with a SIGINT, a brief summary is displayed.

This program is intended for use in network testing, measurement and management. Because of the load it can impose on the network, it is unwise to use ping during normal operations or from automated scripts.

ICMP PACKET DETAILS

An IP header without options is 20 bytes. An ICMP ECHO_REQUEST packet contains an additional 8 bytes worth of ICMP header followed by an arbitrary amount of data. When a packetsize is given, this indicated the size of this extra piece of data (the default is 56). Thus the amount of data received inside of an IP packet of type ICMP ECHO_REPLY will always be 8 bytes more than the requested data space (the ICMP header).

If the data space is at least eight bytes large, ping uses the first eight bytes of this space to include a timestamp

which it uses in the computation of round trip times. If less than eight bytes of pad are specified, no round trip times are given.

DUPLICATE AND DAMAGED PACKETS

Ping will report duplicate and damaged packets. Duplicate packets should never occur, and seem to be caused by inappropriate link-level retransmissions. Duplicates may occur in many situations and are rarely (if ever) a good sign, although the presence of low levels of duplicates may not always be cause for alarm.

Damaged packets are obviously serious cause for alarm and often indicate broken hardware somewhere in the ping packet's path (in the network or in the hosts).

TRYING DIFFERENT DATA PATTERNS

The (inter)network layer should never treat packets differently depending on the data contained in the data portion. Unfortunately, data-dependent problems have been known to sneak into networks and remain undetected for long periods of time. In many cases the particular pattern that will have problems is something that doesn't have sufficient ``transitions'', such as all ones or all zeros, or a pattern right at the edge, such as almost all zeros. It isn't necessarily enough to specify a data pattern of all zeros (for example) on the command line because the pattern that is of interest is at the data link level, and the relationship between what you type and what the controllers transmit can be complicated.

This means that if you have a data-dependent problem you will probably have to do a lot of testing to find it. If you are lucky, you may manage to find a file that either can't be sent across your network or that takes much longer to transfer than other similar length files. You can then examine this file for repeated patterns that you can test using the -p option of ping.

TTL DETAILS

The TTL value of an IP packet represents the maximum number of IP routers that the packet can go through before being thrown away. In current practice you can expect each router in the Internet to decrement the TTL field by exactly one.

The TCP/IP specification states that the TTL field for TCP packets should be set to 60, but many systems use smaller values (4.3BSD uses 30, 4.2 used 15).

The maximum possible value of this field is 255, and most Unix systems set the TTL field of ICMP ECHO_REQUEST packets to 255. This is why you will find you can ``ping'' some

hosts, but not reach them with telnet(1) or ftp(1).

In normal operation ping prints the ttl value from the packet it receives. When a remote system receives a ping packet, it can do one of three things with the TTL field in its response:

+ Not change it; this is what Berkeley Unix systems did before the 4.3BSD-tahoe release. In this case the TTL value in the received packet will be 255 minus the number of routers in the round-trip path.

+ Set it to 255; this is what current Berkeley Unix systems do. In this case the TTL value in the received packet will be 255 minus the number of routers in the path from the remote system to the ping'ing host.

+ Set it to some other value. Some machines use the same value for ICMP packets that they use for TCP packets, for example either 30 or 60. Others may use completely wild values.

BUGS

Many Hosts and Gateways ignore the RECORD_ROUTE option.

The maximum IP header length is too small for options like RECORD_ROUTE to be completely useful. There's not much that that can be done about this, however.

Flood pinging is not recommended in general, and flood pinging the broadcast address should only be done under very controlled conditions.

SEE ALSO

netstat(1), ifconfig(8), routed(8)

HISTORY

The ping command appeared in 4.3BSD.

NAME

pstat - print system facts

SYNOPSIS

/usr/sbin/pstat -aixptufbcmsT [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 a 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.

RSS Size of resident text, measured in multiples of 512 bytes.

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.

FORW Forward link in free list.

BACK Backward link in free list.

-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 (by signal or under trace)

F Miscellaneous state variables, or'ed together (hexadecimal):

0001 loaded
0002 system process (swapper)
0004 locked for swap out
0008 swap save area
0010 traced
0020 used in tracing
0040 user settable lock in core
0100 process resulted from vfork(2)
0200 parent in vfork, waiting for child
0400 parent has released child in vfork
1000 detached inherited by init
2000 no SIGCHLD signal to parent
4000 selecting; wakeup/waiting danger

PRI Scheduling priority, see setpriority(2).

SIG 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.

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.

-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:
T delay timeout in progress
W waiting for open to complete
O open
F outq has been flushed during DMA
C carrier is on
B busy doing output
A process is awaiting output
X open for exclusive use
S output stopped
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. Only the fields located in the first page cluster can be located succesfully if the process is in main memory.

-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
S shared lock present
X exclusive lock present
I signal pgrp when data ready
CNT Number of processes that know this open file.
MSG Number of messages outstanding for this file.
DATA The location of the inode table entry or socket structure for this file.
OFFSET
The file offset (see lseek(2)).

-b Print the buffer pool table with the these headings:

IND Index of buffer descriptor
LOC Memory address of buffer descriptor
FLAGS Miscellaneous state variables encoded thus:
R B_READ: read when I/O occurs
D B_DONE: transaction finished
E B_ERROR: transaction aborted
B B_BUSY: not on av_forw/back list
P B_PHYS: physical IO
M B_MAP: alloc UNIBUS
W B_WANTED: issue wakeup when BUSY goes off
A B_AGE: delayed write for correct aging
a B_ASYNC: don't wait for I/O completion
d B_DELWRI: write at exit of avail list
T B_TAPE: this is a magtape (no bdwrite)
I B_INVAL: does not contain valid info
b B_BAD: bad block revectoring in progress
L B_LOCKED: locked in core (not reusable)
u B_UBAREMAP: addr UNIBUS virtual, not physical
r B_RAMREMAP: remapped into ramdisk
FORW Hash chain forward link (as index)
BACKW Hash chain backward link (as index)
AFORW Alternate chain forward link (as index)
ABACKW
Alternate chain backward link (as index)
DEVICE
Device major, minor number
BLKNO Logical block number on device

-s print the swapmap and a summary on swap space usage. The summary gives the number of used and available swapmap entries, the total amount of allocated and free swap space in kByte, and the size of the largest free swapspace segment.

-c print the coremap and a summary on free memory areas.

The summary gives the number of used and available coremap entries as well as the total amount and size largest segment of free memory in kBytes.

-m print the ub_map, describing free UNIBUS mapping registers, and a summary on free mapping registers. The summary gives the number of used and available ub_map entries as well as the total amount and size largest segment of free registers.

-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

iostat(1), ps(1), systat(1), vmstat(1), stat(2), fs(5), K. Thompson, UNIX Implementation

BUGS

It would be very useful if the system recorded "maximum occupancy" on the tables reported by -T; even more useful if these tables were dynamically allocated.

NAME

quot - summarize file system ownership

SYNOPSIS

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

/etc/passwd to get user names

SEE ALSO

du(1), ls(1)

BUGS

Default file systems vary with installations. Holes in files are counted as if they actually occupied space.

NAME

quotacheck - filesystem quota consistency checker

SYNOPSIS

```
quotacheck [ -v ] filesystem ...
quotacheck [ -v ] -a
```

DESCRIPTION

Quotacheck examines each filesystem, builds a table of current disk usage, and compares this table against that recorded in the disk quota file for the filesystem. 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 filesystem is checked).

Available options:

-a If the -a flag is supplied in place of any filesystem names, quotacheck will check all the filesystems indicated in /etc/fstab to be read-write with disk quotas.

-v quotacheck reports discrepancies between the calculated and recorded disk quotas.

Parallel passes are run on the filesystems required, using the pass numbers in /etc/fstab in an identical fashion to fsck(8).

Normally quotacheck operates silently.

Quotacheck expects each filesystem to be checked to have a quota files named quotas located at the root of the associated file system. These defaults may be overridden in /etc/fstab. If a file is not present, quotacheck will create it.

Quotacheck is normally run at boot time from the /etc/rc.local file, see rc(8), before enabling disk quotas with quotaon(8).

Quotacheck accesses the raw device in calculating the actual disk usage for each user. Thus, the filesystems checked should be quiescent while quotacheck is running.

FILES

quotas at the filesystem root

/etc/fstab default filesystems

BUGS

The quotas file may be named arbitrarily but must reside in the filesystem for which it contains quota information. Quotacheck will give the error:

```
%s dev (0x%x) mismatch %s (0x%x)
```

if the quotas file is not in the filesystem being checked. This restriction is enforced by the kernel but may be lifted in the future.

SEE ALSO

quota(1), quotactl(2), fstab(5), edquota(8), fsck(8), quotaon(8), repquota(8)

HISTORY

The quotacheck command appeared in 4.2BSD.

NAME

quotaon, quotaoff - turn filesystem quotas on and off

SYNOPSIS

```
quotaon  [-v] filesystem ...
quotaon  [-v] -a
quotaoff [-v] filesystem ...
quotaoff [-v] -a
```

DESCRIPTION

Quotaon announces to the system that disk quotas should be enabled on one or more filesystems. Quotaoff announces to the system that the specified filesystems should have any disk quotas diskquotas turned off. The filesystems specified must have entries in /etc/fstab and be mounted. Quotaon expects each filesystem to have a quota file named quotas located at the root of the associated file system. These defaults may be overridden in /etc/fstab.

Available options:

-a If the -a flag is supplied in place of any filesystem names, quotaon/quotaoff will enable/disable all the filesystems indicated in /etc/fstab to be read-write with disk quotas.

-v Causes quotaon and quotaoff to print a message for each filesystem where quotas are turned on or off.

FILES

quotas at the filesystem root with user quotas

/etc/fstab filesystem table

SEE ALSO

quota(1), setquota(2), fstab(5), edquota(8), quotacheck(8), repquota(8)

HISTORY

The quotaon command appeared in 4.2BSD.

NAME

rc - command script for auto-reboot and daemons

SYNOPSIS

/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)

BUGS

NAME

rdump - file system dump across the network

SYNOPSIS

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, /usr/sbin/rmt, on the client machine to access the tape device.

SEE ALSO

dump(8), rmt(8)

DIAGNOSTICS

Same as dump(8) with a few extra related to the network.

NAME

reboot - stopping and restarting the system

SYNOPSIS

/sbin/reboot [-lqnhdarsfRD]
/sbin/halt [-lqndars]
/sbin/fastboot [-lqndarsRD]

DESCRIPTION

2.11BSD is started by placing it in memory at location zero and transferring to its entry point. Since the system is not reentrant, it is necessary to read it in from disk or tape each time it is to be boot strapped.

Rebooting a running system: When the system is running and a reboot is desired, shutdown(8) is normally used to stop time sharing and put the system into single user mode. If there are no users then /sbin/reboot can be used without shutting the system down first.

Reboot normally causes the disks to be synced and allows the system to perform other shutdown activities such as resynchronizing hardware time-of-day clocks. A multi-user reboot (as described below) is then 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 multi-user operation.

Options to reboot are:

-l Don't try to tell syslogd(8) what's about to happen.

-q Reboot quickly and ungracefully, without shutting down running processes first.

-n Don't sync before rebooting. This can be used if a disk or the processor is on fire.

-h Don't reboot, simply halt the processor.

-d Dump memory onto the dump device, usually part of swap, before rebooting. The dump is done in the same way as after a panic.

-a Have the system booter ask for the name of the system to be booted, rather than immediately booting the default system (/unix).

-r Mount the root file system as read only when the system reboots. This is not supported by the kernel in 2.11BSD.

-s Don't enter multi-user mode after system has rebooted - stay in single user mode.

-f Fast reboot. Omit the automatic file system consistency check when the system reboots and goes multi-user. This is accomplished by passing a fast reboot flag on to the rebooting kernel. This currently prevents the use of -f flag in conjunction with the -h (halt) flag.

-D Set the autoconfig(8) debug flag. This is normally not used unless one is debugging the autoconfig program.

-R Tells the kernel to use the compiled in root device. Normally the system uses the device from which it was booted as the root/swap/pipe/dump device.

Reboot normally places a shutdown record in the login accounting file /usr/adm/wtmp. This is inhibited if the -q or -n options are present. Note that the -f (fast reboot) and -n (don't sync) options are contradictory; the request for a fast reboot is ignored in this case.

Halt and fastboot are synonymous with ``reboot -h'' and ``reboot -f'', respectively.

Power fail and crash recovery: Normally, the system will reboot itself at power-up or after crashes if the contents of low memory are intact. An automatic consistency check of the file systems will be performed, and unless this fails, the system will resume multi-user operations.

SEE ALSO

autoconfig(8), sync(2), utmp(8), shutdown(8), syslogd(8)

NAME

renice - alter priority of running processes

SYNOPSIS

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,

```
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_MAX (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_MIN (-20) to PRIO_MAX. Useful priorities are: 20 (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)

BUGS

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.

NAME

repquota - summarize quotas for a file system

SYNOPSIS

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.

NAME

rexecd - remote execution server

SYNOPSIS

/usr/libexec/rexecd

DESCRIPTION

Rexecd is the server for the rexec(3) routine. The server provides remote execution facilities with authentication based on user names and 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, unencrypted 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 initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by rexecd.

DIAGNOSTICS

Except for the last one listed below, all diagnostic messages are returned on the initial socket, 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.

``<shellname>: ...''
The user's login shell could not be started. This message is returned on the connection associated with the stderr, and is not preceded by a flag byte.

SEE ALSO
rexec(3)

BUGS
Indicating ``Login incorrect'' as opposed to ``Password incorrect'' is a security breach which allows people to probe a system for users with null passwords.

A facility to allow all data and password exchanges to be encrypted should be present.

NAME

rlogind - remote login server

SYNOPSIS

rlogind [-aln]

DESCRIPTION

Rlogind is the server for the rlogin(1) program. The server provides a remote login facility with authentication based on privileged port numbers from trusted hosts.

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 512-1023, the server aborts the connection.

2) The server checks the client's source address and requests the corresponding host name (see IR gethost-byaddr (3), hosts(5) and named(8)). If the hostname cannot be determined, the dot-notation representation of the host address is used. If the hostname is in the same domain as the server (according to the last two components of the domain name), or if the -a option is given, the addresses for the hostname are requested, verifying that the name and address correspond. Normal authentication is bypassed if the address verification fails.

Once the source port and address have been checked, rlogind proceeds with the authentication process described in rshd(8). It then 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 -f option if authentication has succeeded. If automatic authentication fails, the user is prompted to log in as if on a standard terminal line. The -l option prevents any authentication based on the user's ``.rhosts'' file, unless the user is logging in as the superuser.

The parent of the login process manipulates the master side of the pseudo 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). The screen or window size of the terminal is requested from the client, and window size changes from the client are propagated to the pseudo terminal.

Transport-level keepalive messages are enabled unless the -n option is present. The use of keepalive messages allows sessions to be timed out if the client crashes or becomes unreachable.

DIAGNOSTICS

All initial diagnostic messages are indicated by a leading byte with a value of 1, after which any network connections are closed. If there are no errors before login is invoked, a null byte is returned as in indication of success.

``Try again.''
A fork by the server failed.

SEE ALSO

login(1), ruserok(3), rshd(8)

BUGS

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.

A facility to allow all data exchanges to be encrypted should be present.

A more extensible protocol should be used.

NAME

rmt - remote magtape protocol module

SYNOPSIS

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(3) or rcmd(3) 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

Anumber\n

where number is an ASCII representation of a decimal number. Unsuccessful commands are responded to with

Eerror-number\nerror-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 Acount-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(3), rexec(3), mtio(4), rdump(8), rrestore(8)

BUGS
People tempted to use this for a remote file access protocol are discouraged.

NAME

route - manually manipulate the routing tables

SYNOPSIS

/sbin/route [-f] [-n] [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(8), should tend to this task.

Route accepts two commands: add, to add a route, and delete, to delete a route.

All commands have the following syntax:

/sbin/route command [net | host] destination gateway [metric]

where destination is the destination host or network, gateway is the next-hop gateway to which packets should be addressed, and metric is a count indicating the number of hops to the destination. The metric is required for add commands; it must be zero if the destination is on a directly-attached network, and nonzero if the route utilizes one or more gateways. If adding a route with metric 0, the gateway given is the address of this host on the common network, indicating the interface to be used for transmission. Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with destination. The optional keywords net and host force the destination to be interpreted as a network or a host, respectively. Otherwise, if the destination has a ``local address part'' of INADDR_ANY, or if the destination is the symbolic name of a network, 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 as a host name using gethostbyname(3N). If this lookup fails, getnetbyname(3N) is then used to interpret the name as that of a network.

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.

The -n option prevents attempts to print host and network names symbolically when reporting actions.

DIAGNOSTICS

``add [host | network] %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. If the gateway address used was not the primary address of the gateway (the first one returned by gethostbyname), the gateway address is printed numerically as well as symbolically.

``delete [host | network] %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.

``Network is unreachable''
An attempt to add a route failed because the gateway listed was not on a directly-connected network. The next-hop gateway must be given.

``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(8), XNSrouted(8)

NAME

routed - network routing daemon

SYNOPSIS

routed [-d] [-g] [-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. It used a generalized protocol capable of use with multiple address types, but is currently used only for Internet routing within a cluster of networks.

In normal operation routed listens on the udp(4) socket for the route service (see services(5)) 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 that 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 updates the kernel routing table. The change is reflected in the next response packet sent.

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 local internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks. The response is sent to the broadcast address on nets capable of that function, to the destination address on point-to-point links, and to the router's own address on other networks. The normal routing tables are bypassed when sending gratuitous responses. The reception of responses on each network is used to determine that the network and interface are functioning correctly. If no response is received on an interface, another route may be chosen to route around the interface, or the route may be dropped if no alternative is available.

Routed supports several options:

-d Enable additional debugging information to be logged, such as bad packets received.

-g This flag is used on internetwork routers to offer a route to the ``default'' destination. This is typically used on a gateway to the Internet, or on a gateway that uses another routing protocol whose routes are not reported to other local routers.

-s Supplying this option forces routed to supply routing information whether it is acting as an internetwork router or not. This is the default if multiple network interfaces are present, or if a point-to-point link is in use.

-q This is the opposite of the -s option.

-t 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, if not tracing all packets, 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 located using only information from 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. External gateways are also passive, but are not placed in the kernel routing table nor are they included in routing updates. The function of external entries is to inform routed that another routing process will install such a route, and that alternate routes to that destination should not be installed. Such entries are only required when both routers may learn of routes to the same destination.

The /etc/gateways is comprised of a series of lines, each in the following format:

< net | host > name1 gateway name2 metric value < passive | active | external >

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, if started after named(8), known to the name server), or an Internet address specified in ``dot'' notation; see inet(3).

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.

One of the keywords passive, active or external indicates if the gateway should be treated as passive or active (as described above), or whether the gateway is external to the scope of the routed protocol.

Internetwork routers that are directly attached to the Arpanet or Milnet should use the Exterior Gateway Protocol (EGP) to gather routing information rather then using a static routing table of passive gateways. EGP is required in order to provide routes for local networks to the rest of the Internet system. Sites needing assistance with such configurations should contact the Computer Systems Research Group at Berkeley.

FILES

/etc/gateways for distant gateways

SEE ALSO

``Internet Transport Protocols'', XSIS 028112, Xerox System Integration Standard.
udp(4), XNSrouted(8), htable(8)

BUGS

The kernel's routing tables may not correspond to those of routed when redirects change or add routes. The only remedy for this is to place the routing process in the kernel.

Routed should incorporate other routing protocols, such as Xerox NS (XNSrouted(8)) and EGP. Using separate processes for each requires configuration options to avoid redundant or competing routes.

Routed should listen to intelligent interfaces, such as an IMP, and to error protocols, such as ICMP, to gather more information. It does not always detect unidirectional failures in network interfaces (e.g., when the output side fails).

NAME

rrestore - restore a file system dump across the network

SYNOPSIS

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, rmt, on the client machine to access the tape device.

SEE ALSO

restore(8), rmt(8)

DIAGNOSTICS

Same as restore(8) with a few extra related to the network.

NAME

rshd - remote shell server

SYNOPSIS

rshd [-aln]

DESCRIPTION

Rshd is the server for the rcmd(3) routine and, consequently, for the rsh(1) program. The server provides remote execution facilities with authentication based on privileged port numbers from trusted hosts.

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 512-1023, 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 2 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 512-1023.

4) The server checks the client's source address and requests the corresponding host name (see gethostbyaddr(3), hosts(5) and named(8)). If the hostname cannot be determined, the dot-notation representation of the host address is used. If the hostname is in the same domain as the server (according to the last two components of the domain name), or if the -a option is given, the addresses for the hostname are requested, verifying that the name and address correspond. If address verification fails, the connection is aborted with the message, ``Host address mismatch.''

5) 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.

6) 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.

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 using ruserok(3), which uses the file ``/etc/hosts.equiv'' and the ``.rhosts'' file found in the user's home directory. The -l option prevents ruserok(3) from doing any validation based on the user's ``.rhosts'' file, unless the user is the superuser.

9) A null byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by rshd.

Transport-level keepalive messages are enabled unless the -n option is present. The use of keepalive messages allows sessions to be timed out if the client crashes or becomes unreachable.

DIAGNOSTICS

Except for the last one listed below, all diagnostic messages are returned on the initial socket, 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 execution of the login shell).

``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).

``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.

``Can't fork; try again.''
A fork by the server failed.

``<shellname>: ...''
The user's login shell could not be started. This message is returned on the connection associated with the stderr, and is not preceded by a flag byte.

SEE ALSO

rsh(1), rcmd(3), ruserok(3)

BUGS

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.

A facility to allow all data exchanges to be encrypted should be present.

A more extensible protocol (such as Telnet) should be used.

NAME

rwhod - system status server

SYNOPSIS

/usr/sbin/rwhod

DESCRIPTION

Rwhod is the server which maintains the database used by the rwho(1) and ruptime(1) 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 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 {
            structoutmp 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; they are multiplied by 100 for representation in an integer. The host name included is that returned by the gethostname(2) system call, with any trailing domain name omitted. 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 in seconds since a character was last received on the terminal line.

Messages received by the rwho server are discarded unless they originated at an 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 3 minutes. Rwhod performs an nlist(3) on /vmunix every 30 minutes to guard against the possibility that this file is not the system image currently operating.

SEE ALSO

rwho(1), ruptime(1)

BUGS

There should be a way to relay status information between networks. Status information should be sent only upon request rather than continuously. People often interpret the server dying or network communtication failures as a machine going down.

NAME

rxformat - format floppy disks (2.11BSD)

SYNOPSIS

rxformat 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, where ``?'' is either "a" or "b" to indicate single or double density access. The ``raw'' device must be used. 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 if standard input is a tty (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/rrx??

SEE ALSO

rx(4)

AUTHOR

Helge Skrivervik

BUGS

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.)

NAME

sa - system accounting

SYNOPSIS

sa [-abcdDfijkKlnrstu] [-v threshold] [-S savacctfile] [-U usracctfile] [file]

DESCRIPTION

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 Print all command names, even those containing unprintable characters and those used only once. By default, those are placed 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.

S The following filename is used as the command summary file instead of /usr/adm/savacct.

U The following filename is used instead of /usr/adm/usracct to accumulate the per-user statistics printed by the -m option.

FILES

/usr/adm/acct	raw accounting
/usr/adm/savacct	summary
/usr/adm/usracct	per-user summary

SEE ALSO

ac(8), accton(8), acctd(8)

BUGS

The number of options to this program is absurd.

NAME

savecore - save a core dump of the operating system

SYNOPSIS

savecore dirname [system]

DESCRIPTION

Savecore is meant to be called at the end of the /etc/rc file. Its function is to save the core dump of the system (if one was made) and to write a reboot message in the shutdown log.

It saves the core image in the file dirname/core.n and its corresponding namelist in dirname/unix.n. The second argument is the namelist for the system which made the core image; the current system is always assumed to be /unix. 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 file system 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 savecore detects that the system time is wrong because of a crash (the time in the core image is after the current time), it will reset the system time to its best estimate of the time, which is the time in the core image plus the elapsed time since the reboot. It announces the time that it set when this occurs.

FILES

/usr/adm/shutdownlogshutdown log
/unix current UNIX

BUGS

The method used to determine whether a dump is present, and to prevent the same core image from being saved multiple times, is not elegant. This information should be passed to init by the system; however, this is difficult because the system may have to be rebooted a second time if the root filesystem is patched.

NAME

sendmail - send mail over the internet

SYNOPSIS

/usr/sbin/sendmail [flags] [address ...]

newaliases

mailq [-v]

DESCRIPTION

Sendmail sends a message to one or more recipients, 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 an end-of-file or a line consisting only of a single dot and sends a copy of the message found there to all of the addresses listed. It determines the network(s) 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. Sendmail will fork and run in background listening on socket 25 for incoming SMTP connections. This is normally run from /etc/rc.

-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 on standard input and output. 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.

-Cfile Use alternate configuration file. Sendmail refuses to run as root if an alternate configuration file is specified. The frozen configuration file is bypassed.

-dX Set debugging value to X.

-Ffullname Set the full name of the sender.

-fname Sets the name of the ``from'' person (i.e., the sender of the mail). -f can only be used by ``trusted'' users (normally root, daemon, and network) or if the person you are trying to become is the same as the person you are.

-hN 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. If not specified, ``Received:'' lines in the message are counted.

-n Don't do aliasing.

-oxvalue Set option x to the specified value. Options are described below.

-q[time] Processed saved messages in the queue at given intervals. If time is omitted, process the queue once. Time 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. If time is specified, sendmail will run in background. This option can be used safely with -bd.

-rname An alternate and obsolete form of the -f flag.

-t Read message for recipients. To:, Cc:, and Bcc: lines will be scanned for recipient addresses. The Bcc: line will be deleted before transmission. Any addresses in the argument list will be suppressed, that is, they will not receive copies even if listed in the message header.

-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 Sendmail Installation and Operation Guide. The options are:

Afile Use alternate alias file.

c On mailers that are considered ``expensive'' to connect to, don't initiate immediate connection. This requires queueing.

dx 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.

ex 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.

Fmode The mode to use when creating temporary files.

f Save UNIX-style From lines at the front of messages.

gN The default group id to use when calling mailers.

Hfile The SMTP help file.

i Do not take dots on a line by themselves as a message terminator.

Ln 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.

Qqueuedir Select the directory in which to queue messages.

rtimeout The timeout on reads; if none is set, sendmail will wait forever for a mailer. This option violates the word (if not the intent) of the SMTP specification, show the timeout should probably be fairly large.

Sfile Save statistics in the named file.

s Always instantiate the queue file, even under circumstances where it is not strictly necessary. This provides safety against system crashes during delivery.

Ttime Set the timeout on undelivered messages in the queue to the specified time. After delivery has failed (e.g., because of a host being down) for this amount of time, failed messages will be returned to the sender. The default is three days.

tstz,dtz Set the name of the time zone.

uN Set the default user id for mailers.

In aliases, the first character of a name may be a vertical bar to cause interpretation of the rest of the name as a command to pipe the mail to. It may be necessary to quote the name to keep sendmail from suppressing the blanks from between arguments. For example, a common alias is:

```
   msgs: "|/usr/ucb/msgs -s"
```

Aliases may also have the syntax ``:include:filename'' to ask sendmail to read the named file for a list of recipients. For example, an alias such as:

```
   poets: ":include:/usr/local/lib/poets.list"
```

would read /usr/local/lib/poets.list for the list of addresses making up the group.

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 /etc/sendmail.cf, these pathnames are all specified in /etc/sendmail.cf. Thus, these values are only approximations.

/etc/aliases	raw data for alias names
/etc/aliases.pag	
/etc/aliases.dir	data base of alias names
/etc/sendmail.cf	configuration file
/etc/sendmail.fc	frozen configuration
/usr/share/misc/sendmail.hf	help file
/var/log/sendmail.st	collected statistics
/usr/spool/mqueue/*	temp files

SEE ALSO

mail(1), rmail(1), syslog(3), aliases(5), sendmail.cf(5), mailaddr(7), rc(8);
DARPA Internet Request For Comments RFC819, RFC821, RFC822;
Sendmail - An Internetwork Mail Router (SMM:16);
Sendmail Installation and Operation Guide (SMM:7)

NAME

shutdown - close down the system at a given time

SYNOPSIS

shutdown [-k] [-r] [-h] [-f] [-n] 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 system log, containing the time of shutdown, who ran shutdown and the reason. Then a terminate signal is sent to 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!)

With the -f option, shutdown arranges, in the manner of fastboot(8), that when the system is rebooted the file systems will not be checked. The -n option prevents the normal sync(2) before stopping.

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

SEE ALSO

login(1), reboot(8), fastboot(8)

BUGS

Only allows you to kill the system between now and 23:59 if you use the absolute time for shutdown.

NAME

slattach - attach serial lines as network interfaces

SYOPNSIS

slattach ttyname [baudrate]

DESCRIPTION

Slattach is used to assign a tty line to a network interface, and to define the network source and destination addresses. The ttyname parameter is a string of the form ``ttyXX'', or ``/dev/ttyXX''. The optional baudrate parameter is used to set the speed of the connection. If not specified, the default of 9600 is used.

Only the super-user may attach a network interface.

To detach the interface, use `ifconfig interface-name down' after killing off the slattach process. interface-name is the name that is shown by netstat(1)

EXAMPLES

```
slattach ttyh8
slattach /dev/tty01 4800
```

DIAGNOSTICS

Messages indicating the specified interface does not exit, the requested address is unknown, the user is not privileged and tried to alter an interface's configuration.

SEE ALSO

rc(8), intro(4), netstat(1), ifconfig(8)

NAME

sticky - persistent text and append-only directories

DESCRIPTION

The sticky bit (file mode bit 01000, see chmod(2)) is used to indicate special treatment for certain executable files and directories.

STICKY TEXT EXECUTABLE FILES

While the `sticky bit' 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. Shareable text segments are normally placed in a least-frequently-used cache after use, and thus the `sticky bit' has little effect on commonly-used text images.

Sharable executable files are made by the -n and -z options of ld(1).

Only the super-user can set the sticky bit on a sharable executable file.

STICKY DIRECTORIES

A directory whose `sticky bit' is set becomes an append-only directory, or, more accurately, a directory in which the deletion of files is restricted. A file in a sticky directory may only be removed or renamed by a user if the user has write permission for the directory and the user is the owner of the file, the owner of the directory, or the super-user. This feature is usefully applied to directories such as /tmp which must be publicly writable but should deny users the license to arbitrarily delete or rename each others' files.

Any user may create a sticky directory. See chmod(1) for details about modifying file modes.

BUGS

Since the text areas of sticky text executables are stashed in the swap area, abuse of the feature can cause a system to run out of swap.

Neither open(2) nor mkdir(2) will create a file with the sticky bit set.

NAME

swapon - specify additional device for paging and swapping

SYNOPSIS

```
swapon -a
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
```

BUGS

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.

swapon is not implemented in 2.11BSD.

NAME

sync - update the super block

SYNOPSIS

sync

DESCRIPTION

Sync executes the sync system primitive. Sync can be called to insure that all disk writes have been completed before the processor is halted in a way not suitably done by reboot(8) or halt(8). Generally, it is preferable to use reboot or halt to shut down the system, as they may perform additional actions such as resynchronizing the hardware clock and flushing internal caches before performing a final sync.

See sync(2) for details on the system primitive.

SEE ALSO

sync(2), fsync(2), halt(8), reboot(8), update(8)

NAME

sync - update super-block

SYNOPSIS

sync()

DESCRIPTION

Sync causes all information in core memory that should be on disk to be written out. This includes modified super blocks, modified i-nodes, and delayed block I/O.

Sync should be used by programs that examine a file system, for example fsck, df, etc. Sync is mandatory before a boot.

SEE ALSO

fsync(2), sync(8), update(8)

BUGS

The writing, although scheduled, is not necessarily complete upon return from sync.

NAME

syslogd - log systems messages

SYNOPSIS

syslogd [-fconfigfile] [-mmarkinterval] [-d]

DESCRIPTION

Syslogd reads and logs messages into a set of files described by the configuration file /etc/syslog.conf. Each message is one line. A message can contain a priority code, marked by a number in angle braces at the beginning of the line. Priorities are defined in <sys/syslog.h>. Syslogd reads from the UNIX domain socket /dev/log, from an Internet domain socket specified in /etc/services, and from the special device /dev/klog (to read kernel messages).

Syslogd configures when it starts up and whenever it receives a hangup signal. Lines in the configuration file have a selector to determine the message priorities to which the line applies and an action. The action field are separated from the selector by one or more tabs.

Selectors are semicolon separated lists of priority specifiers. Each priority has a facility describing the part of the system that generated the message, a dot, and a level indicating the severity of the message. Symbolic names may be used. An asterisk selects all facilities. All messages of the specified level or higher (greater severity) are selected. More than one facility may be selected using commas to separate them. For example:

```
*.emerg;mail,daemon.crit
```

Selects all facilities at the emerg level and the mail and daemon facilities at the crit level.

Known facilities and levels recognized by syslogd are those listed in syslog(3) without the leading ``LOG_''. The additional facility ``mark'' has a message at priority LOG_INFO sent to it every 20 minutes (this may be changed with the -m flag). The ``mark'' facility is not enabled by a facility field containing an asterisk. The level ``none'' may be used to disable a particular facility. For example,

```
*.debug;mail.none
```

Sends all messages except mail messages to the selected file.

The second part of each line describes where the message is to be logged if this line is selected. There are four forms:

+ A filename (beginning with a leading slash). The file will be opened in append mode.

+ A hostname preceeded by an at sign (``@''). Selected messages are forwarded to the syslogd on the named host.

+ A comma separated list of users. Selected messages are written to those users if they are logged in.

+ An asterisk. Selected messages are written to all logged-in users.

Blank lines and lines beginning with `#' are ignored.

For example, the configuration file:

```
kern,mark.debug /dev/console
*.notice;mail.info    /usr/spool/adm/syslog
*.crit                 /usr/adm/critical
kern.err        @ucbarpa
*.emerg         *
*.alert         eric,kridle
*.alert;auth.warning  ralph
```

logs all kernel messages and 20 minute marks onto the system console, all notice (or higher) level messages and all mail system messages except debug messages into the file /usr/spool/adm/syslog, and all critical messages into /usr/adm/critical; kernel messages of error severity or higher are forwarded to ucbarpa. All users will be informed of any emergency messages, the users ``eric'' and ``kridle'' will be informed of any alert messages, and the user ``ralph'' will be informed of any alert message, or any warning message (or higher) from the authorization system.

The flags are:

-f Specify an alternate configuration file.

-m Select the number of minutes between mark messages.

-d Turn on debugging.

Syslogd creates the file /var/run/syslog.pid, if possible, containing a single line with its process id. This can be used to kill or reconfigure syslogd.

To bring syslogd down, it should be sent a terminate signal (e.g. kill `cat /var/run/syslog.pid`).

FILES

/etc/syslog.conf the configuration file

/var/run/syslog.pid	the process id
/dev/log	Name of the UNIX domain datagram log socket
/dev/klog	The kernel log device

SEE ALSO

logger(1), syslog(3)

NAME

talkd - remote user communication server

SYNOPSIS

/usr/libexec/ntalkd

DESCRIPTION

Talkd is the server that notifies a user that somebody else wants to initiate a conversation. It acts a repository of invitations, responding to requests by clients wishing to rendezvous to hold a conversation. In normal operation, a client, the caller, initiates a rendezvous by sending a CTL_MSG to the server of type LOOK_UP (see <protocols/talkd.h>). This causes the server to search its invitation tables to check if an invitation currently exists for the caller (to speak to the callee specified in the message). If the lookup fails, the caller then sends an ANNOUNCE message causing the server to broadcast an announcement on the callee's login ports requesting contact. When the callee responds, the local server uses the recorded invitation to respond with the appropriate rendezvous address and the caller and callee client programs establish a stream connection through which the conversation takes place.

SEE ALSO

talk(1), write(1)

NAME

telnetd - DARPA TELNET protocol server

SYNOPSIS

/usr/libexec/telnetd

DESCRIPTION

Telnetd is a server which supports the DARPA standard TELNET virtual terminal protocol. Telnetd is invoked by the internet server (see inetd(8)), normally for requests to connect to the TELNET port as indicated by the /etc/services file (see services(5)).

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 remote client and the login process.

When a TELNET session is started up, telnetd sends TELNET options to the client side indicating a willingness to do remote echo of characters, to suppress go ahead, and to receive terminal type information from the remote client. If the remote client is willing, the remote terminal type is propagated in the environment of the created login process. The pseudo-terminal allocated to the client is configured to operate in "cooked" mode, and with XTABS and CRMOD enabled (see tty(4)).

Telnetd is willing to do: echo, binary, suppress go ahead, and timing mark. Telnetd is willing to have the remote client do: binary, terminal type, and suppress go ahead.

SEE ALSO

telnet(1C)

BUGS

Some TELNET commands are only partially implemented.

The TELNET protocol allows for the exchange of the number of lines and columns on the user's terminal, but telnetd doesn't make use of them.

Because of bugs in the original 4.2 BSD telnet(1C), telnetd performs some dubious protocol exchanges to try to discover if the remote client is, in fact, a 4.2 BSD telnet(1C).

Binary mode has no common interpretation except between similar operating systems (Unix in this case).

The terminal type name received from the remote client is converted to lower case.

The packet interface to the pseudo-terminal (see pty(4)) should be used for more intelligent flushing of input and output queues.

Telnetd never sends TELNET go ahead commands.

NAME

tftpd - DARPA Trivial File Transfer Protocol server

SYNOPSIS

tftpd [directory ...]

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). The server is normally started by inetd(8).

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. Files may be written only if they already exist and are publicly writable. 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. The server should have the user ID with the lowest possible privilege.

Access to files may be restricted by invoking tftpd with a list of directories by including pathnames as server program arguments in /etc/inetd.conf. In this case access is restricted to files whose names are prefixed by the one of the given directories.

SEE ALSO

tftp(1), inetd(8)

NAME

timed - time server daemon

SYNOPSIS

timed [-t] [-M] [-n network] [-i network]

DESCRIPTION

Timed is the time server daemon and is normally invoked at boot time from the rc(8) file. It synchronizes the host's time with the time of other machines in a local area network running timed(8). These time servers will slow down the clocks of some machines and speed up the clocks of others to bring them to the average network time. The average network time is computed from measurements of clock differences using the ICMP timestamp request message.

The service provided by timed is based on a master-slave scheme. When timed(8) is started on a machine, it asks the master for the network time and sets the host's clock to that time. After that, it accepts synchronization messages periodically sent by the master and calls adjtime(2) to perform the needed corrections on the host's clock.

It also communicates with date(1) in order to set the date globally, and with timedc(8), a timed control program. If the machine running the master crashes, then the slaves will elect a new master from among slaves running with the -M flag. A timed running without the -M flag will remain a slave. The -t flag enables timed to trace the messages it receives in the file /usr/adm/timed.log. Tracing can be turned on or off by the program timedc(8). Timed normally checks for a master time server on each network to which it is connected, except as modified by the options described below. It will request synchronization service from the first master server located. If permitted by the -M flag, it will provide synchronization service on any attached networks on which no current master server was detected. Such a server propagates the time computed by the top-level master. The -n flag, followed by the name of a network which the host is connected to (see networks(5)), overrides the default choice of the network addresses made by the program. Each time the -n flag appears, that network name is added to a list of valid networks. All other networks are ignored. The -i flag, followed by the name of a network to which the host is connected (see networks(5)), overrides the default choice of the network addresses made by the program. Each time the -i flag appears, that network name is added to a list of networks to ignore. All other networks are used by the time daemon. The -n and -i flags are meaningless if used together.

FILES

/usr/adm/timed.log	tracing file for timed
/usr/adm/timed.masterlog	log file for master timed

SEE ALSO

date(1), adjtime(2), gettimeofday(2), icmp(4P), timedc(8), TSP: The Time Synchronization Protocol for UNIX 4.3BSD, R. Gusella and S. Zatti

NAME

timedc - timed control program

SYNOPSIS

timedc [command [argument ...]]

DESCRIPTION

Timedc is used to control the operation of the timed program. It may be used to:

+ measure the differences between machines' clocks,

+ find the location where the master time server is running,

+ enable or disable tracing of messages received by timed, and

+ perform various debugging actions.

Without any arguments, timedc will prompt for commands from the standard input. If arguments are supplied, timedc interprets the first argument as a command and the remaining arguments as parameters to the command. The standard input may be redirected causing timedc to read commands from a file. Commands may be abbreviated; recognized commands are:

? [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.

clockdiff host ...
Compute the differences between the clock of the host machine and the clocks of the machines given as arguments.

trace { on | off }
Enable or disable the tracing of incoming messages to timed in the file /usr/adm/timed.log.

quit
Exit from timedc.

Other commands may be included for use in testing and debugging timed; the help command and the program source may be consulted for details.

FILES

/usr/adm/timed.log tracing file for timed

/usr/adm/timed.masterlog log file for master timed

SEE ALSO

date(1), adjtime(2), icmp(4P), timed(8),
TSP: The Time Synchronization Protocol for UNIX 4.3BSD, R. Gusella and S. Zatti

DIAGNOSTICS

?Ambiguous command abbreviation matches more than one command
?Invalid command no match found
?Privileged command command can be executed by root only

NAME

trpt - transliterate protocol trace

SYNOPSIS

trpt [-a] [-s] [-t] [-f] [-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.

-a in addition to the normal output, print the values of the source and destination addresses for each packet recorded.

-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.

-f follow the trace as it occurs, waiting a short time for additional records each time the end of the log is reached.

-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, the address of which follows.

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. The -f option can be used to follow the trace log once the trace is located. 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. The

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), trsp(8C)

DIAGNOSTICS

``no namelist'' when the system image doesn't contain the proper symbols to find the trace buffer; others which should be self explanatory.

BUGS

Should also print the data for each input or output, but this is not saved in the race record.

The output format is inscrutable and should be described here.

NAME

trsp - transliterate sequenced packet protocol trace

SYNOPSIS

trsp [-a] [-s] [-t] [-j] [-p hex-address] [system [core]]

DESCRIPTION

Trpt interrogates the buffer of SPP 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, trsp prints all the trace records found in the system grouped according to SPP 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 trsp 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 trsp 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.

BUGS

Should also print the data for each input or output, but this is not saved in the race record.

The output format is inscrutable and should be described here.

NAME

tunefs - tune up an existing file system

SYNOPSIS

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.

-o optimization preference
The file system can either try to minimize the time spent allocating blocks, or it can attempt minimize the space fragmentation on the disk. If the value of minfree (see above) is less than 10%, then the file system should optimize for space to avoid running out of full sized blocks. For values of minfree greater than or equal to 10%, fragmentation is unlikely to be problematical, and the file system can be optimized for time.

SEE ALSO

fs(5), newfs(8), mkfs(8)

M. McKusick, W. Joy, S. Leffler, R. Fabry, ``A Fast File System for UNIX'', ACM Transactions on Computer Systems 2, 3. pp 181-197, August 1984. (reprinted in the System Manager's Manual, SMM:14)

BUGS

This program should work on mounted and active file systems. Because the super-block is not kept in the buffer cache, the changes will only take effect if the program is run on dismounted file systems. To change the root file system, the system must be rebooted after the file system is tuned.

tunefs is not currently implemented in 2.11BSD.

You can tune a file system, but you can't tune a fish.

NAME

update - periodically update the super block

SYNOPSIS

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, but should be executed out of the initialization shell command file.

SEE ALSO

sync(2), sync(8), init(8), rc(8)

BUGS

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.

NAME

uucico, uucpd - transfer files queued by uucp or uux

SYNOPSIS

/usr/sbin/uucico [-dspooldir] [-ggrade] [-rrole] [-R] [-ssystem] [-xdebug] [-L] [-tturnaround]

/usr/libexec/uucpd

DESCRIPTION

Uucico performs the actual work involved in transferring files between systems. Uucp(1) and uux(1) merely queue requests for data transfer which uucico processes.

The following options are available.

-dspooldir
Use spooldir as the spool directory. The default is /usr/spool/uucp.

-ggrade Only send jobs of grade grade or higher this transfer. The grade of a job is specified when the job is queued by uucp or uux.

-rrole role is either 1 or 0; it indicates whether uucico is to start up in master or slave role, respectively. 1 is used when running uucico by hand or from cron(8). 0 is used when another system calls the local system. Slave role is the default.

-R Reverse roles. When used with the -r1 option, this tells the remote system to begin sending its jobs first, instead of waiting for the local machine to finish.

-ssystem
Call only system system. If -s is not specified, and -r1 is specified, uucico will attempt to call all systems for which there is work. If -s is specified, a call will be made even if there is no work for that system. This is useful for polling.

-xdebug Turn on debugging at level debug. Level 5 is a good start when trying to find out why a call failed. Level 9 is very detailed. Level 99 is absurdly verbose. If role is 1 (master), output is normally written to the standard message output stderr. If stderr is unavailable, output is written to /usr/spool/uucp/AUDIT/system. When role is 0 (slave), debugging output is always written to the AUDIT file.

-L Only call ``local'' sites. A site is considered local if the device-type field in L.sys is one of LOCAL, DIR or TCP.

-tturnaround
Use turnaround as the line turnaround time (in minutes) instead of the default 30. If turnaround is missing or 0, line turnaround will be disabled. After uucico has been running in slave role for turnaround minutes, it will attempt to run in master role by negotiating with the remote machine. In earlier versions of uucico, a transfer of many large files in one direction would hold up mail going in the other direction. With the turnaround code working, the message flow will be more bidirectional in the short term. This option only works with newer uucico's and is ignored by older ones.

If uucico receives a SIGFPE (see kill(1)), it will toggle the debugging on or off.

Uucpd is the server for supporting uucp connections over networks. Uucpd listens for service requests at the port indicated in the ``uucp'' service specification; see services(5). The server provides login name and password authentication before starting up uucico for the rest of the transaction.

Uucico is commonly used either of two ways: as a daemon run periodically by cron(8) to call out to remote systems, and as a ``shell'' for remote systems who call in. For calling out periodically, a typical line in crontab would be:

```
0 * * * * /usr/sbin/uucico -r1
```

This will run uucico every hour in master role. For each system that has transfer requests queued, uucico calls the system, logs in, and executes the transfers. The file L.sys(5) is consulted for information about how to log in, while L-devices(5) specifies available lines and modems for calling.

For remote systems to dial in, an entry in the passwd(5) file must be created, with a login ``shell'' of uucico. For example:

```
nuucp:Password:6:1::/usr/spool/uucppublic:/usr/sbin/uucico
```

The UID for UUCP remote logins is not critical, so long as it differs from the UUCP Administrative login. The latter owns the UUCP files, and assigning this UID to a remote login would be an extreme security hazard.

FILES

/etc/uucp/	UUCP internal files
/etc/uucp/L-devices	Local device descriptions
/etc/uucp/L-dialcodes	Phone numbers and prefixes
/etc/uucp/L.aliases	Hostname aliases
/etc/uucp/L.cmds	Remote command permissions list
/etc/uucp/L.sys	Host connection specifications
/etc/uucp/USERFILE	Remote directory tree permissions list
/usr/spool/uucp/	Spool directory
/usr/spool/uucp/AUDIT/*	Debugging audit trails
/usr/spool/uucp/C./	Control files directory
/usr/spool/uucp/D./	Incoming data file directory
/usr/spool/uucp/D.hostname/	Outgoing data file directory
/usr/spool/uucp/D.hostnameX/	Outgoing execution file directory
/usr/spool/uucp/CORRUPT/	Place for corrupted C. and D. files
/usr/spool/uucp/ERRLOG	UUCP internal error log
/usr/spool/uucp/LOGFILE	UUCP system activity log
/usr/spool/uucp/LCK/LCK..*	Device lock files
/usr/spool/uucp/SYSLOG	File transfer statistics log
/usr/spool/uucp/STST/*	System status files
/usr/spool/uucp/TM./	File transfer temp directory
/usr/spool/uucp/X./	Incoming execution file directory
/usr/spool/uucppublic	Public access directory

SEE ALSO

uucp(1), uuq(1), uux(1), L-devices(5), L-dialcodes(5), L.aliases(5), L.cmds(5), L.sys(5), uuclean(8), uupoll(8), uusnap(8), uuxqt(8)

D. A. Nowitz and M. E. Lesk, A Dial-Up Network of UNIX Systems.

D. A. Nowitz, Uucp Implementation Description.

NAME

uuclean - uucp spool directory clean-up

SYNOPSIS

/etc/uucp/uuclean [-m] [-ntime] [-ppre]

DESCRIPTION

Uuclean will scan the spool directory for files with the specified prefix and delete all those which are older than the specified number of hours.

The following options are available.

-ppre Scan for files with pre as the file prefix. Up to 10 -p arguments may be specified.

-ntime Files whose age is more than time hours will be deleted if the prefix test is satisfied. (default time is 72 hours)

-m Send mail to the owner of the file when it is deleted.

-dsubdirectory
Only the specified subdirectory will be cleaned.

This program will typically be run daily by cron(8).

FILES

/usr/spool/uucp Spool directory

SEE ALSO

uucp(1), uux(1), uucico(8)

NAME

uupoll - poll a remote UUCP site

SYNOPSIS

uupoll [-ggrade] [-n] system

DESCRIPTION

Uupoll is used to force a poll of a remote system. It queues a null job for the remote system and then invokes uucico(8).

The following options are available:

-ggrade Only send jobs of grade grade or higher on this call.

-n Queue the null job, but do not invoke uucico.

Uupoll is usually run by cron(5) or by a user who wants to hurry a job along. A typical entry in crontab could be:

```
0    0,8,16     *    *    *     /usr/bin/uupoll ihnp4
0    4,12,20    *    *    *     /usr/bin/uupoll ucbvax
```

This will poll ihnp4 at midnight, 0800, and 1600, and ucbvax at 0400, noon, and 2000.

If the local machine is already running uucico every hour and has a limited number of outgoing modems, a more elegant approach might be:

```
0    0,8,16    *    *    * /usr/bin/uupoll -n ihnp4
0    4,12,20   *    *    * /usr/bin/uupoll -n ucbvax
5    *         *    *    * /usr/sbin/uucico -r1
```

This will queue null jobs for the remote sites at the top of hour; they will be processed by uucico when it runs five minutes later.

FILES

/etc/uucp/ UUCP internal files
/usr/spool/uucp/ Spool directory

SEE ALSO

uucp(1), uux(1), uucico(8)

NAME

uusnap - show snapshot of the UUCP system

SYNOPSIS

uusnap

DESCRIPTION

Uusnap displays in tabular format a synopsis of the current UUCP situation. The format of each line is as follows:

```
        site   N Cmds   N Data     N Xqts      Message
```

Where "site" is the name of the site with work, "N" is a count of each of the three possible types of work (command, data, or remote execute), and "Message" is the current status message for that site as found in the STST file.

Included in "Message" may be the time left before UUCP can re-try the call, and the count of the number of times that UUCP has tried (unsuccessfully) to reach the site.

SEE ALSO

uucp(1C), uux(1C), uuq(1C), uucico(8C)
UUCP Implementation Guide

NAME

uuxqt - UUCP execution file interpreter

SYNOPSIS

/usr/libexec/uuxqt [-xdebug]

DESCRIPTION

Uuxqt interprets execution files created on a remote system via uux(1) and transferred to the local system via uucico(8). When a user uses uux to request remote command execution, it is uuxqt that actually executes the command. Normally, uuxqt is forked from uucico to process queued execution files; for debugging, it may also be run manually by the UUCP administrator.

Uuxqt runs in its own subdirectory, /usr/spool/uucp/XTMP. It copies intermediate files to this directory when necessary.

FILES

/etc/uucp/L.cmds	Remote command permissions list
/etc/uucp/USERFILE	Remote directory tree permissions list
/usr/spool/uucp/LOGFILE	UUCP system activity log
/usr/spool/uucp/LCK/LCK.XQT	Uuxqt lock file
/usr/spool/uucp/X./	Incoming execution file directory
/usr/spool/uucp/XTMP	Uuxqt running directory

SEE ALSO

uucp(1), uux(1), L.cmds(5), USERFILE(5), uucico(8)

NAME

vipw - edit the password file

SYNOPSIS

vipw

DESCRIPTION

Vipw edits the password file while setting the appropriate locks, and does 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 entries, and will not allow a password file with a ``mangled'' entry to be installed. If vipw rejects the new password file, the user is prompted to re-enter the edit session.

Once the information has been verified, vipw uses mkpasswd(8) to update the user database. This is run in the background, and, at very large sites could take several minutes. Until this update is completed, the password file is unavailable for other updates and the new information will not be available to programs.

SEE ALSO

chpass(1), passwd(1), passwd(5), adduser(8), mkpasswd(8)

NAME

XNSrouted - NS Routing Information Protocol daemon

SYNOPSIS

/sbin/XNSrouted [-s] [-q] [-t] [logfile]

DESCRIPTION

XNSrouted is invoked at boot time to manage the Xerox NS routing tables. The NS routing daemon uses the Xerox NS Routing Information Protocol in maintaining up to date kernel routing table entries.

In normal operation XNSrouted listens for routing information packets. If the host is connected to multiple NS networks, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When XNSrouted 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. XNSrouted 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, XNSrouted 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 XNSrouted 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, XNSrouted 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, XNSrouted 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 to other routers.

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 XNSrouted 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, XNSrouted 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 XNSrouted'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.

SEE ALSO

``Internet Transport Protocols'', XSIS 028112, Xerox System Integration Standard.
idp(4P)