NAME

intro − introduction to UNIX system explanations

DESCRIPTION

This section explains the procedures for operating the UNIX system itself. Included are procedures for reboot, crash recovery, loading UNIX, and "panic" error message handling.

NAME

boot procedures − UNIX startup

DESCRIPTION

In the Unlikely (?) Event of a System Crash.

1) Mount a scratch tape with the write ring in place on tape drive as unit 0.

2) Lower Halt Key.

3) Load Address 44 by placing swtiches 5 and 2 up, all others down and depressing and releasing the Load Address Key.

4) Raise Halt Key and depress Start Key. Observe tape motion.

5) When tape stops lower Halt Key and manually rewind and unload tape. Give tape to a SYSTEM GURU.

6) Follow reboot procedures described below.

To Halt System

1) Login as a UNIX user authorized to execute the telinit command on the system TTY.

2) Use the "wall" command to warn all users to log off the system.

3) When the "who" command indicates that you are the only user on the system, type

```
telinit 7
```

followed by a carriage return.

4) After waiting about 30 seconds (to allow all programs to terminate) type

```
sync
```

followed by a carriage return and when prompt is returned lower the halt key (BEWARE: the processor key should be in the "power" position; if in the "lock" position the processor will not halt).

To Reboot System When Halted

1) Insure that the disk is online.

2) Place 17765000 in the CSW and depress and release the Load Address Key.

3) Place an octal 0 in the CSW to boot from the default boot device.

4) Raise Halt Key and depress and release Start Key. System will respond with

```
hpboot:
```

5) Type

```
unix
```

followed by a carriage return if the standard UNIX is to be booted. If for some reason the standard UNIX is not functioning properly, unix.old, which is usually a more seasoned version of unix, may be typed for this step instead of unix.

6) The system will then prompt for the new runlevel. If this does not occur, depress the interrupt (delete) key to obtain the prompt. If the system has just crashed and the superuser password is known enter a 7 followed by a carriage return and proceed with step 7 below. Otherwise enter the multiuser run level for your system (usually 0) and proceed with step 8 below.

(PERFORM STEP 7 IF SYSTEM CRASHED AND ROOT PASSWORD IS KNOWN)

7) The system will now prompt for the root password (if one exists). Enter the password followed by a carriage return. If the password you enter is incorrect the system will reprompt for a new runlevel. Enter the multiuser level at this time. If the password you enter is correct, the

system will respond with a pound sign ("#"). Type

```
                    /etc/mount
```

followed by a carriage return to determine what was mounted at the time of the crash. Check those file systems indicated by the mount command as well as the root file system (/dev/rootdev) by typing

```
                    check rootdev x y z ...
```

where x y z ... are the names reported by the mount command. If any errors are reported and you are unsure of their exact meaning do *not* proceed, but find a SYSTEM GNOME to fix the problem. If any errors on the root file system are corrected it will be necessary to halt the processor *without* utilizing the sync command and repeat this procedure starting at step one. This should be done when indicated by the check command. After checking the file systems type

```
                    telinit n
```

followed by a carriage return, where *n* is the multiuser runlevel for your system.

8) The system will then prompt for information for the boot history file. After this information has been gathered the system should enter the desired runlevel.

The switches. During operation, the console switches are examined 60 times per second, and the contents of the address specified by the switches are displayed in the display register. If the switch address is even, the address is interpreted in kernel (system) space; if odd, the rounded-down address is interpreted in the current user space.

FILES

/unix − UNIX code

SEE ALSO

tp(1), init(7)

NAME

crash − what to do when the system crashes

DESCRIPTION

This section gives at least a few clues about how to proceed if the system crashes. It can't pretend to be complete.

How to bring it back up. If the reason for the crash is not evident (see below for guidance on 'evident'), you may want to try to dump the system if you feel up to debugging. Currently a dump can be taken only on magtape. With a tape mounted and ready on drive 0, stop the machine, load address 44, and start. 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. See below for what to do with the tape.

In restarting after a crash, always bring up the system single-user. This is accomplished by following the directions in *boot procedures* (6) as modified for your particular installation. When it is running, perform a *check*(1M) on all file systems which could have been in use at the time of the crash. If any serious file system problems are found, they should be repaired. When you are satisfied with the health of your disks, check and set the date if necessary, then come up multi-user.

To boot UNIX at all, four files (and the directories leading to them) must be intact. First, the initialization program */etc/init* must be present and executable. If it is not, the CPU will loop in user mode at location 6. For *init* to work correctly, */dev/syscon*, */bin/sh*, and */bin/su* must be present. If any do not exist, *init* will loop trying to create a Shell with proper standard input and output.

If you cannot get the system to boot, a runnable system must be obtained from a backup medium. The root file system may then be doctored as a mounted file system as described below. If there are any problems with the root file system, it is probably prudent to go to a backup system to avoid working on a running file system.

Repairing disks. The first rule to keep in mind is that an added disk should be treated gently; it shouldn't be mounted unless necessary. If it is very valuable, yet in bad shape, it should be dumped before trying surgery. This is an area where experience and informed courage count for much.

The problems reported by *check* typically fall into two kinds. There can be problems with the free list: duplicates in the free list, or free blocks also in files. These can be cured easily with a *check −s*. There can also be problems if the same block appears in more than one file or if a file contains bad blocks, the files should be deleted, and the free list reconstructed. The best way to delete such a file is to use *clri* (1M), then remove its directory entries. If any of the affected files is really precious, you can try to copy it to another device first.

Check may report files which have more directory entries than links. Such situations are potentially dangerous; *clri* discusses a special case of the problem. All the directory entries for the file should be removed. If on the other hand there are more links than directory entries, there is no danger of spreading infection, but merely some disk space that is lost for use. It is sufficient to copy the file (if it has any entries and is useful); then use *clri* on its inode and remove any directory entries that do exist.

Finally, there may be inodes reported by *check* that have 0 links and 0 entries. These occur on the root device when the system is stopped with pipes open, and on other file systems when the system stops with files that have been deleted while still open. A *clri* will free the inode, and an *check −s* will recover any missing blocks.

Why did it crash? UNIX types a message on the console typewriter when it voluntarily crashes. See *uemess*(6) for a description of the possible messages generated.

Interpreting dumps. All file system problems should be taken care of before attempting to look at dumps. The dump should be read into a file; *cp(1)* will do. At this point, you should execute *ps −alxk* and *who* to print the process table and the users who were on at the time of the crash. You should dump (*od*(1)) the first 30 bytes of the dump file. Starting at location 4, the registers R0, R1, R2, R3, R4, R5, SP and KDSA6 (KISA6 for 11/40s) are stored. If the dump had to be restarted, R0 will not be correct. Next, take the value of KA6 (location 22(8) in the dump) multiplied by 100(8) and dump 1000(8) bytes starting from there. This is the per-process data associated with the process running at the time of the crash. Relabel the addresses 140000 to 141776. 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 until you obtain an R5 value of 141756, which is where the user's R5 is stored. If the chain is broken, you have to look for a plausible R5, PC pair and continue from there. Each PC should be looked up in the system's name list, using *adb(1)* and its ':' command to get a reverse calling order. In most cases this procedure will give an idea of what is wrong. A more complete discussion of system debugging is impossible here.

SEE ALSO

clri(1M), check(1M), stack(1M)
"Explanation of Abnormal Conditions within the UNIX Operating System",MMF,3/17/75.

NAME

Unix Error Messages − description of UNIX console messages

GENERAL DESCRIPTION

Error messages directed to the system console, **/dev/syscon**, may originate from at least three sources: the UNIX operating system proper; the initialization program, *init*; or *getty* , the program responsible for the initial setup of a teletype line. This document lists all messages which may emanate from any of these sources, briefly describes the condition causing the message, and gives a recommended action to alleviate the problem.

UNIX SYSTEM MESSAGES

Messages which originate within the operating system itself are of two types: panic messages and informative messages. Informative messages usually indicate that a correctable error has occurred and that the system will attempt to continue processing. Panic messages, on the other hand, are a result of an error condition within the system which precludes further processing. Thus, those messages described below which begin with the string **panic:** result in an attempt by the system to gracefully halt processing. In these cases it is necessary to reboot the system to resume.

Some of the ACTION sections below will optionally recommend the generation of a dump tape. If it is desired to do this a good tape with the write ring inserted should be mounted and placed online on tape drive unit zero. Halt the processor, load address 44, and start. The dump routine will then copy all of memory to tape followed by an EOF. The tape will *not* rewind: this must be done manually.

bad block on dev *major/minor*

DESCRIPTION:

Printed if a block in a file's block list is too large or small for the file system containing the file. The file is on the major/minor device specified by the message.

ACTION:

Determine which file system is bad by comparing the major/minor device numbers in the message to the major/minor devices for all mountable file systems. Utilize the *ls* command to do this. When the offending file system has been determined, unmount and check the file system before referencing it again. If the offending file system is the root file system it will be necessary to change to single user mode, check the file system, and reboot.

bad count on dev *major/minor*

DESCRIPTION:

Printed if certain magic numbers in the superblock of a mounted file system are not reasonable. The file system is on the major/minor device specified by the message.

ACTION:

Perform the same ACTION as that described for the **bad block** error message.

DANGER: mfree map overflow *nnn* **lost** *mmm* **items at** *xxx*

DESCRIPTION:

The resource map at octal address *nnn* was full and could not accept another fragment. *mmm* items of the resource were lost starting at *xxx*. The system will probably continue to run in a degraded fashion.

ACTION:

Reboot as soon as possible. Determine which resource map has the address *nnn* and enlarge the map.

DMC11(*num*): block not found

DESCRIPTION:

Occurs if the DMC11 finishes an I/O operation on a block not on the list of active blocks. This can only occur if the DMC11 has a hardware malfunction or if some other device is malfunctioning severely enough to cause the DMC11 buffer queues to be overwritten. *Num* is the number of the DMC11 which is malfunctioning.

ACTION:

If the DMC11 itself is malfunctioning, more messages will be forthcoming; the best action is to wait to see if the problem recurs. If many messages are forthcoming, the DMC11 should be serviced.

gettm: no free fde

DESCRIPTION:

The logical file handler was unable to seize an in-core file defintion entry (fde) for a logical file access because the list of available fde's is empty.

ACTION:

None. The process that is trying to access the logical file will alternately scan and sleep until a free fde is seized or another process secures an fde for the same logical file. This is mainly a debugging message and not an error message.

******hpioctl: All drives disabled.**

******hpioctl: All drives enabled.**

DESCRIPTION:

These messages are generated whenever a hpioctl is done by the super user. If drives are enabled(default), the system will behave normally. If disabled, all real time activities required by hardware interrupts will occur normally, but no disk IO requests will be honored. The hpioctl was implemented primarily for dual-ported disk systems where it is desired to change the position of the drive controller switch without causing a crash or halting the system.

ACTION:

These messages are informational only and do not indicate an error condition.

hpstrategy: unit out of range

DESCRIPTION:

This message occurs whenever a request is made to queue a disk job for a disk whose minor device(unit) number is greater than the value defined for **NHP** in param.h.

ACTION:

Fix the software which made the invalid request or change the value of **NHP** as appropriate.

Inode table overflow

DESCRIPTION:

This message occurs when an attempt to open a new file fails due to the in-core inode table being full.

ACTION:

If the message persists it may indicate that the size of the inode table is too small. This can only be fixed by increasing the table size, regenerating the operating system, and booting the new system. See also the ACTION for the **no file** message.

no file

DESCRIPTION:

Printed if an attempt to open a file fails due to the file table being full.

ACTION:

If the message persists the system parameter governing the size of the file table should be increased, the system regenerated and then rebooted. Note that the size of the file table and inode table are related: the size of inode table governs the maximum number of uniquely different files which can be simultaneously accessed by all the programs in the system; the size of the file table governs the maximum number of open files which can simultaneously exist in the system. That is to say that an inode table slot is allocated whenever a file is opened and that file has not yet been opened by any other process, whereas a file table slot is allocated whenever an open is performed for any file. Thus, since one file may be opened by more than one process simultaneously, the size of the file table should always be greater than the size of the inode table. Alternatively, one can state that there is an entry in the file table for each instance of an open file with two significant exceptions: file descriptors which have been created by means of the *dup*(2) system call or inherited via the *fork*(2) system call do not allocate an additional file table slot.

no space on dev *major/minor*

DESCRIPTION:

There are no blocks remaining for file storage on the file system specified by *major/minor*.

ACTION:

See the **bad block** message ACTION to determine how to find the logical name of the offending file system; then remove some unnecessary files from the file system.

Out of inodes on dev *major/minor*

DESCRIPTION:

The system attempted to allocate an inode on a file system for which there were no free inodes. This message does not imply that the file system is "full"; rather, it indicates that no new files or directories may be created.

ACTION:

Remove unnecessary files from the file system or remake the file system with more blocks allocated to the inode list by utilizing the *mkfs*(1) UNIX command.

out of text

DESCRIPTION:

This message occurs whenever the system attempts to allocate a text segment for a new process and finds that the text table is full. The new process is not created.

ACTION:

Decrease the number of different text shared programs running simultaneously or increase the text table size, remake and reboot the system.

panic: blkdev

DESCRIPTION:

This message is printed when a request is made for one of the system buffers to do I/O to a bad logical device (bad major device number specification). Bad logical device numbers should have been rejected at higher levels of system software before reaching the I/O subsystem.

ACTION:
If problem persists generate a dump tape for a UNIX guru. In all cases a reboot is necessary.

panic: bmap

DESCRIPTION:
This message occurs if the system discovers that a file or directory which it was changing from type small to large was, in fact, no longer of type file or directory.

ACTION:
Generate a dump tape for a UNIX guru. Reboot.

panic: buffers

DESCRIPTION:
This occurs at boot time if the system is unable to allocate enough memory for the external buffers.

ACTION:
Regenerate the system with fewer external buffers.

panic: devtab

DESCRIPTION:
If a device driver attempts to allocate a system buffer and does not have a devtab entry in the block device switch table this **panic** message is generated.

ACTION:
This message generally implies that the block device switch table was incorrectly or incompletely specified at sysgen time. This table should be checked and corrected and the system remade and rebooted. If this is not the problem a dump tape should be generated for the local UNIX guru.

panic: iinit

DESCRIPTION:
This message is printed during UNIX startup if the root file system cannot be mounted. It is the result of a bad specification of the root device at sysgen time or a hardware problem during boot time such as the device being offline.

ACTION:
Correct the problem. Regenerate the system if necessary. Reboot.

panic: IO err in swap

DESCRIPTION:
This message is generated whenever a attempt is made to read or write the swap device results in a noncorrectable error.

ACTION:
Insure that the swap device is powered up, online, and write enabled. If this is a new system insure that the swap device has been correctly specified at sysgen time. The three parameters of significance are **swapdev**, **swaplo**, and **nswap**. These variables correspond to the major/minor device, the lowest number logical block on this device to be used for swapping, and the number of blocks to be used for swapping, respectively. A peculiarity of **swaplo** is that UNIX has built into it the notion that block 0 of any logical device cannot be allocated. Therefore, **swaplo** should never be set to zero.

panic: no clock

DESCRIPTION:
This message is generated if, when UNIX is booted, no clock can be located on the

UNIBUS.

ACTION:
Verify that the clock is functional and present on the UNIBUS at the standard address.

panic: no fs

DESCRIPTION:
Occurs when the system cannot find the superblock for a particular device by searching the system's mount table.

ACTION:
Generate a dump tape for a UNIX guru. Reboot.

panic: no imt

DESCRIPTION:
Occurs when a file on a file system which is mounted is referenced but the system is unable to find the file system in the mount table.

ACTION:
Generate a dump tape for a UNIX guru. Reboot.

panic: no msgmem

DESCRIPTION:
This message occurs at boot time if the system is unable to allocate enough memory for messages.

ACTION:
Regenerate the system with less message space.

panic: no procs

DESCRIPTION:
Printed if, during the spawning of a new process, the system finds that it is out of room in the process table when it knows due to a previous check that there is room in the table.

ACTION:
Generate a dump tape for a UNIX guru. Reboot.

panic: maus

DESCRIPTION:
This message occurs at boot time if the system is unable to allocate enough memory for multiply accessable user space.

ACTION:
Regenerate the system with fewer or smaller MAUS segments.

panic: out of swap space

DESCRIPTION:
This message occurs whenever an attempt is made to allocate swap space and none is available.

ACTION:
If the problem persists, increase the number of blocks in the swap area, regenerate the system, and reboot the new system.

panic: putfs

DESCRIPTION:
Occurs when the system attempts to decrement the reference count for a superblock that it cannot find in the mount table.

ACTION:
Generate a dump tape for a UNIX guru. Reboot.

panic: Running a dead proc

DESCRIPTION:
This message is generated if the system determines that a dead process is about to be marked runable.

ACTION:
Generate a dump tape for a UNIX guru. Reboot.

panic: Sleeping on wchan 0

DESCRIPTION:
This message is generated if an internal error causes an attempt to suspend a process on wait channel zero.

ACTION:
Generate a dump tape for a UNIX guru. Reboot.

panic: stack overflow

DESCRIPTION:
This message indicates that the system stack overflowed.

ACTION:
Generate a dump tape for a UNIX guru. Reboot. The problem is most likely caused by an excessive nesting of interrupts. If the problem reoccurs and there is no hardware fault it may be necessary to increase the size of the ublock.

panic: Timeout table overflow

DESCRIPTION:
This message will occur whenever an entry needs to be made in the timeout table but the table is full.

ACTION:
If problem persists, increase the size of the timeout table, regenerate the system and reboot.

ka6 = *n*
aps = *m*
trap type *x*
panic: trap

DESCRIPTION:
This message indicates that an unexpected processor trap has occured. The trap can result from an illegal memory reference *by the operating system*, the execution of an illegal instruction *by the operating system*, etc. More likely, it can indicate hardware problems. The octal numbers *n* and *m* are the values of kernel address register 6 and the stack address of the old ps, respectively. These numbers can be used to help debug a system dump. The octal number *x* in the **trap type** message indicates which type of trap occured. The expected values for these numbers are:

0	bus error trap vectored through 04
1	illegal instruction trap vectored through 010
2	bpt-trace trap vectored through 014
3	iot trap vectored through 020
5	emulator trap vectored through 030

6	sys trap vectored through 034
11	segmentation violation trap vectored through 0250

Any other number indicates an unknown trap probably through some nonstandard address.

ACTION:

If the problem persists suspect hardware problems. Certain traps can be generated by new device drivers which have not been completely debugged. Investigate any new drivers recently added to the system, correct any addressing errors found and reboot. This message will also occur if a hardware unit that is not on the UNIBUS is referenced.

parity
lerr=*nnn* **herr**=*nnn* **mserr**=*nnn* **mcr**=*nnn*
panic: parity (optionally)

DESCRIPTION:

This message occurs whenever a memory parity error is detected. The four octal numbers printed out are the values contained in the memory system registers located at the UNIBUS addresses 017777740, 017777742, 017777744, and 017777746 at the time of the error. If the error occurs within the operating system the optional **panic** occurs.

ACTION:

If the error results in a **panic: parity** message it is wise to clear the error from the console before attempting to reboot. Using **lerr** and **herr**, determine the address of the word in memory containing the parity error and deposit a zero into this location from the console. See the 11/70 Processor Handbook for a description of how to interpret the registers. If problem persists, consult DEC.

panic: x25buffers

DESCRIPTION:

The x25buffers panic in x25b.c occurs only when there is not enough memory to allocate space for the special BX.25 buffer area at system initialization time. The panic cannot occur on a normally running, multi-user system (i.e., ONLY when main is calling init routines, before even init is run).

ACTION:

reconfigure system

panic: xclist

DESCRIPTION:

This message occurs at boot time if the system is unable to allocate enough memory for the external clist.

ACTION:

Regenerate the system with a smaller external clist.

pcs: getsampbuf err

DESCRIPTION:

Unable to acquire a system buffer to be used for profiling.

ACTION:

Try again.

nnn
*** POWER FAIL RESTART ***

DESCRIPTION:

This message occurs when the power is reapplied to a UNIX system that has been powered down or taken a power hit. The message indicates that the operating system is attempting to restart. The number *nnn* is the number of machine cycles that remained during powerdown after the powerfail save sequence was complete.

ACTION:

Certain I/O devices may not come back online by themselves after a power failure. In particular, turn any DEC line printers back online and restart any programs using the mag tape.

proc on q

DESCRIPTION:

This cryptic message is printed if the system attempts to add a process to the run queue which it discovers is already on the run queue.

ACTION:

If the problem persists the message should be changed to a **panic** and a core dump should be generated.

Recursive panic!: *cause*

DESCRIPTION:

This message is printed when, while processing one **panic** another occurs. *Cause* is the second panic message.

ACTION:

Generate a dump tape for a UNIX guru. Reboot.

RP04/5/6 drive *num* **offline**

DESCRIPTION:

This message is printed whenever an RP04, RP05, or RP06 drive is switched offline.

ACTION:

If this message occurs when no manual action has been taken at the drive a hardware problem is indicated. Whenever this message is generated all pending I/O for the offline drive is flushed.

RP04/5/6 Disk Drive(s) *n n* ... **offline. Manual Attention reqired.**

DESCRIPTION:

This message is printed whenever an RP04, RP05, or RP06 drive does not come back online after a power failure. The message will come out every 2 minutes until the offending drive is place back online.

ACTION:

Insure that the drives specified are powered on and online.

samp rlse err: only *xx* **of** *yy*

DESCRIPTION:

All of the system buffers used for profiling were not released back to the system. (*yy* system buffers were acquired for profiling and only *xx* of them were given back when finished).

ACTION:

Reboot is needed to recover lost system buffers.

stray interrupt at *addr*

DESCRIPTION:

This message is printed whenever an unexpected interrupt occurs through the octal address *addr*.

ACTION:

If this message persists the offending hardware should be isolated and repaired.

system **RELEASE** *release.issue*
REAL MEM = *nnn* **BYTES**
AVAIL MEM = *mmm* **BYTES**

DESCRIPTION:

This message is generated whenever UNIX is booted. *System* normally prints out as **CB-UNIX**. *Release.issue* is currently **2.3**. *nnn* is the number of bytes that the operating system thinks are physically present. *mmm* is the number of bytes available for user programs.

ACTION:

If the number of bytes reported as being present is less than that which is expected some memory is not responding when its address is placed on the bus and DEC should be consulted.

x25scan: nobuf

DESCRIPTION:

The "x25scan: nobuf" message occurs on receive buffer exhaustion. This can occur as new links are installed while existing links are using all the available buffers. If at configuration time a proper ratio of buffer space to number of links exists this situation should never occur.

ACTION:

reconfigure system

INIT MESSAGES

Messages originating from the system init program, /etc/**init**, are easily distinguishable since as indicated below they are all preceded by the string **INIT:**. These messages are generally the result of an error in the controlling lines file /**etc/inittab**. For a description of the *init* program and *inittab* file see section 1 and 5 of the manual, respectively.

INIT: execlp of /bin/su failed; errno = *nn*

DESCRIPTION:

In trying to go into single user mode, *init* was unable to exec /**bin/su**.

ACTION:

Replace /**bin/su** .

INIT: Internal process table is full.

DESCRIPTION:

The internal table in which *init* keeps track of its children is full and so a new process cannot be created.

ACTION:

Decrease the number of active entries in *inittab* file or increase init's process table size. The latter requires recompiling *init* and rebooting.

INIT: Command is respawning too rapidly. Check for possible errors.
id:*cc command*

DESCRIPTION:

The command in *inittab* with the id *cc* has died more than **ten** times in two minutes. *Init* generates this message on the assumption that a process dying that often is probably in error, and put a **five** minute suspension on the execution of this command to give the operator a chance to correct the problem.

ACTION:

Two things can cause this problem. A typo in *inittab* or a program which has been removed accidentally and is referenced in *inittab*. Correct the typo or replace the program. To terminate the five minute suspension after the problem is corrected, type " **telinit Q** ".

INIT: Command
/bin/sh *command* **failed to execute. errno =** *nn*

DESCRIPTION:

The exec to /**bin/sh** failed for some reason.

ACTION:

Most likely reason is that /**bin/sh** has disappeared. Replace it.

INIT: Cannot open /etc/inittab

DESCRIPTION:

/etc/**inittab** has disappeared.

ACTION:

Replace /etc/**inittab**.

INIT: SINGLE USER MODE

DESCRIPTION:

Init has just changed to the single user (level 7) mode of operation.

ACTION:

This message is the normal result of changing to the single user mode.

INIT: Cannot fork – errno = *nn*

DESCRIPTION:

Due to a system overload condition *init* is unable to generate a new process.

ACTION:

Either reduce the number of processes in the system or generate a new operating system with the capacity to manage a larger number of processes.

GETTY MESSAGES

The line initialization program, /**etc/getty**, is frequently invoked by *init* to initialize teletype lines. The messages below may be generated by *getty* if an error is encountered.

getty: no terminal line specified.

DESCRIPTION:

Getty was invoked with no arguments and consequently does not know what line to open.

ACTION:

Insure that the lines file entries which explicitly specify the **getty** program pass it at least one argument, namely, the line *getty* is to open.

getty: unable to find *speed* **in** /etc/**gettydefs.**

DESCRIPTION:

Getty was unable to find the speed definition field in /etc/**gettydefs**.

ACTION:
Make sure that the speed argument is typed correctly and appears in in /etc/gettydefs. (See *gettydefs*(5) if a new definition is required.)

getty: *linedisc* is an undefined line discipline.

DESCRIPTION:
An invalid line discipline was passed to *getty*.

ACTION:
Make sure name of line discipline is correct. (See *gettydefs*(5) for legal line discipline names. If a new line discipline is added, *getty* needs to be modified and recompiled.)

getty: pointer to next speed in entry *speed* is bad.

DESCRIPTION:
The /etc/gettydefs file is bad. In particular, when trying to autobaud to a new speed, *getty* detected that the next speed it was supposed to go to wasn't listed in /etc/gettydefs.

ACTION:
Correct /etc/gettydefs .

getty: unable to find *speed* again.

DESCRIPTION:
After the preceeding error was encountered, *getty* discovered that it could not find the speed entry it came from in /etc/gettydefs .

ACTION:
Most likely that /etc/gettydefs has been scribbled or has been removed. Correct or replace it.

getty: cannot open *tty*.

DESCRIPTION:
The specified *tty* cannot be opened. In all likelihood it doesn't exist in the directory /dev.

ACTION:
Correct typo in /etc/inittab or make node for the *tty*.

getty: can't open /etc/gettydefs.

DESCRIPTION:
The file /etc/gettydefs couldn't be opened.

ACTION:
Make sure that /etc/gettydefs file exists.