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 e vident (see below for guidance on 'evident') you may want to try to dump the system if you feel up to debugging. At the moment a dump can be taken only on magtape. With a tape mounted and ready, stop the machine, load address 44, and start. This should write a copy of all of core on the tape with an EOF mark.

In restarting after a crash, always bring up the system single-user. This is accomplished by following the directions in Operations Manual; a single-user system is indicated by having a particular value in the switches (173030 unless you've changed *init)* as the system starts executing. When it is running, perform a*c heck(VIII)* 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. This is most easily accomplished by changing the single-user value in the switches to something else, then logging out by typing an EOT.

To even boot UNIX at all, three 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/tty8* and */bin/sh* must be present. If either does not exist, the symptom is best described as thrashing. *Init* will go into a *fork/e xec* 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 mounted file system.

Repairing disks. The first rule to k eep in mind is that an addled disk should be treated gently; it shouldn't be mounted unless necessary, and if it is very valuable yet in quite bad shape, perhaps it should be dumped before trying surgery on it.

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 an *icheck −s[drive type],* where the drive type corresponds to either the number 3 or 4 (RPO*3,* RPO*4).* If the same block appears in more than one file or if a file contains bad blocks, the files can be deleted, and the free list reconstructed. The best way to delete a file containing bad blocks is to use *clrm(VIII),* then remove its directory entries. Files that contain duplicate blocks are done differently. All the file's i-numbers that reference the duplicate block are first identified by *check −b [block #] device.* Their names can then be found by *ncheck −i* and removed after the file system is mounted.

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 *clrm* will free the inode, and an *icheck −s[drive type]* will recover any missing blocks.

Why did it crash? UNIX types a message on the console terminal when it voluntarily crashes. Here is the current list of such messages, with enough information to provide a hope at least of the remedy. The message has the form 'panic: ...', possibly accompanied by other information. Left unstated in all cases is the possibility that hardware or software error produced the message in some unexpected way.

blkdev

The *getblk* routine was called with a nonexistent major device as argument. Definitely hardware or software error.

devtab
: Null device table entry for the major device used as argument to *getblk*. Definitely hardware or software error.

timeout table overflow
: Too many entries in timeout table.

iinit
: An I/O error reading the super-block for the root file system during initialization.

no fs
: A device has disappeared from the mounted-device table. Definitely hardware or software error.

no imt
: Like ‘no fs’, but produced elsewhere.

no clock
: During initialization, neither the line nor programmable clock was found to exist.

I/O error in swap
: An unrecoverable I/O error during a swap. Really shouldn’t be a panic, but it is hard to fix.

out of swap space
: A program needs to be swapped out, and there is no more swap space. It has to be increased. This really shouldn’t be a panic, but there is no easy fix.

trap
: An unexpected trap has occurred within the system. This is accompanied by three numbers: a ‘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; and a ‘trap type’ which encodes which trap occurred. The trap types are:

  0 bus error
  1 illegal instruction
  2 BPT/trace
  3 IOT
  4 power fail
  5 EMT
  6 recursive system call (TRAP instruction)
  7 programmed interrupt request
  8 floating point trap
  9 segmentation violation

In some of these cases it is possible for octal 20 to be added into the trap type; this indicates that the processor was in user mode when the trap occurred. If you wish to examine the stack after such a trap, either dump the system, or use the console switches to examine core; the required address mapping is described below.

Interpreting dumps. All file system problems should be taken care of before attempting to look at dumps. The dump should be read into the file */sys/sys/core; cp*(I) 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*(I)) the first 30 bytes of */sys/sys/core.* Starting at location 4, the registers R0, R1, R2, R3, R4, R5, SP and KDSA6 (KISA6 for 11/40s) are stored. Next, take the value of KA6 (location 22(8) in the dump) multiplied by 100(8) and dump USIZE 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. 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. Each PC should be looked up in the system’s name list using *db*(I) and its ‘:’ command, to get a reverse calling order. In most cases this procedure will give an idea of what is wrong.

SEE ALSO

check(VIII), dcheck(VIII), icheck(VIII), ncheck(VIII), clri(VIII), clrm(VIII), fsdb(VIII)
Repairing Damaged PWB/UNIX File Systems by P. D. Wandzilak
PWB/UNIX Operations Manual by M. E. Pearlman