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.