Crash and Dump Analysis [7]

This chapter includes information on system crash analysis and recovery, including diagnosis and debugging of system problems.

Warning: This chapter contains warnings and information critical to the use of a Cray ML-Safe configuration of the UNICOS operating system.

This chapter does not answer all questions or solve all problems. If you need assistance in debugging a system crash, or if you have a recurring UNICOS kernel problem, contact Software Technical Support at (612) 683-5600. When you call, it is important to have available dumps and a copy of the UNICOS operating system that caused the crash.

7.1 Introduction

This chapter describes system crash analysis and recovery, including the dump process and the operation of the UNICOS kernel debugger `crash`(8). It also discusses ways to diagnose and debug system problems.

When recovering from system crashes, you should have the following information available:

- UNICOS kernel source code, if available
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022
- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011

7.2 Using the `crash` program

The `crash`(8) command is an interactive program that helps locate the problem causing the error by examining the UNICOS system image, which can be either the image of the running system or the image saved following a system crash. `crash` provides an interactive interface that formats and displays system control information from the system memory image.

After you enter the `crash` command, you can examine the following elements of the UNICOS system memory image:

- Buffer and buffer headers
- Callout table
- File table
- I-node table
- Map tables (`coremap` only)
- Buffers
- Mount table
- Process table
- Pseudo-tty table and tty table
- Other system tables
- Stack dump, traceback, and frame package formatting
- System (dump) statistics
- User structures

To use `crash` effectively, you need a dump, knowledge of the computer system instruction set, and familiarity with the UNICOS system kernel. Alternatively, you can run `crash` on an active system by using `/dev/mem` instead of a dump.

Many commands are available in the `crash` utility; those described in this manual are `proc`, `stat`, `trace`, `stack`, and `ut`. Two especially valuable features of `crash` are pipes and redirection, which allow `crash` to interface easily with other UNICOS commands.

Once inside the `crash` utility, you can enter `?` or `help` to produce an abbreviated list of commands. Refer to the `crash`(8) man page for the syntax and description of each command.

Caution: System dumps and the data obtained and written when executing the `crash` command should be labeled at `syshigh` on UNICOS MLS and Cray ML-Safe system configurations to avoid accidental disclosure of protected information.

7.3 Analyzing system problems

This section is a guide to identifying problems that lead to a crash, not a step-by-step guide for debugging crashes. Experience in debugging and knowledge of the UNICOS kernel program are the keys to success in debugging a crash.

System crashes can be divided into the two following categories:

- Panic
- Running system

7.3.1 Panic

The UNICOS kernel program panics when it encounters an unrecoverable hardware or software problem. When UNICOS panics, a panic message is displayed on the operator's workstation. The `SYSDUMP1?` prompt is displayed and the operator is given the opportunity to perform a system dump.

For information on system panics on CRAY EL systems, refer to *CRAY EL Series IOS Messages*, Cray Research publication SQ–2402.

7.3.1.1 Debugging panics

To debug a panicked system, follow these steps:

1. Use the `stat` command (see `crash`(8)), which tells you why the system panicked, the time of the crash, how long the system was up, and the system name. Debugging concentrates on the CPU that detected the panic condition. This information is displayed by `stat`.
2. Executing the `stat` command reports the process slot of the panicked process. For example, the output of the `stat` command may include the following:

   ```
   Panic process: p[76]
   ```

   You can use this process number as the argument to the `stack` command, as in the following example:

   ```
   stack 76
   ```

   Executing `stack` in this manner produces the kernel stack traceback for the panicked process.

3. Use the `ut` command (see `crash`(8)) to examine the system trace buffer. The trace buffer provides a history of the most recent action within the UNICOS system. Try to correlate the trace buffer entries to the data on the stack.
4. Check the load on the system by using the `proc` command (see `crash`(8)) to display all of the current processes. Pay particular attention to the number of swapped processes or large numbers of similarly named processes.

The stack traceback or the trace buffer should have pointed you to the failing area within UNICOS. If the stack data appears to be correct, check the per-process data in the process table and user area for the crashed CPU.

Beyond this point, the number of causes of a panic are too numerous to list. Your best strategy is to use `crash`(8) to examine the details of the data structures in the failing area of the system and to compare the data values for both a running system and the UNICOS kernel source code, if available.

Caution: Security administrators should examine system dumps for security audit data that may have been generated, but not written to the official security log. The administrator can examine the dump via `crash`(8) using `slog` and `rslog`, and by using the `reduce`(8) command.

7.3.1.2 Buffer flushing

The UNICOS operating system provides a feature that minimizes file system corruption by flushing buffered data to its target I/O devices before the system is halted. You implement this feature as a option configurable at compile-time. By default, it is enabled at configuration. To disable the feature, define the `FLUSHONPANIC` variable in the `config.h` file as `0` and compile the kernel. You may do this manually or through the install tool.

When this feature is enabled, three types of buffered data are flushed:

- Kernel data structures that must be updated on disk, such as mount, inode, and quota table entries
- UNICOS system buffer cache
- Logical device cache (`ldcache`)

This feature significantly reduces the amount of user and system data lost when a UNICOS system crashes as a result of a call to the `panic`() routine. The reduction of lost data minimizes file system corruption.

7.3.2 Running system

This section discusses systems that have not crashed, but are not running normally. You may still be able to log into the system, or there may be some users who are still logged in and are able to use the system effectively.

Sometimes it is impossible to fix a system, even when it is still running. If this is the case, log off as many users as possible, and issue the `sync`(1) and `ldsync`(8) commands. When you halt the system, get a dump so that the problem can be studied and solved. This is an extreme measure, so it should not be done without first exhausting all measures for fixing the system while it is still active.

Perform the following steps when the system is not running normally:

1. Examine the system console for error messages.
2. If you are able to log in to the system, use the `ps`(1) command with the `-elf` options, as follows, to get a list of all processes on the system:

   ```
   ps -elf
   ```

   Check for the following conditions:

   - Swapped processes. If you notice a large number of processes swapped out (the `PC_LOAD` bit in per-process flags is off), try to determine what is causing the swapping. Several large processes can cause the system to begin swapping, which degrades the response time for interactive processes.
   - Running processes. A large number of running processes should not degrade interactive response time, but they can degrade turnaround time for utilities that are CPU-intensive, such as compilers, assemblers, and pattern-matching tools.
   - Zombie processes. If you notice more than a few zombie processes, look at the `PGRP` label of the processes in question to determine what happened to the parent process. If the parent is `init`, use `crash` to check the `WCHAN` address (`ds` address) and try to determine why `init` is not waiting for its child processes.

7.4 The `fdmp` command

The `fdmp`(8) command formats system dump images. It also provides the ability to format data from the IOS, which is not available from the `crash`(8)

command. In addition, `fdmp` allows 132-column output suitable for line printers.