

IRIX/SysV-R4 Crash Analysis

By: Micah Altman, (Copyright 1998)

Last Revision: Mar. 1, 1998

All right, nobody panic!

When the system finds itself in a state where continuing would be dangerous to data, it panics -- writes a memory image, and halts. By examining the memory image, we may be able to find the cause of the problem that caused the crash.

On the other hand, if the system panics infrequently, it may be a better allocation of time to simply reboot and forget it. Even if the panics recur, the best approach may be to install all current patches and see if the panic goes away.

- *What happens during and after a crash?*

- The system writes a memory image to the swap partition. Kernel data is written first, then user pages, and finally pages that are thought to be unused. If the memory image is larger than the partition, some information may be lost.

Under IRIX, if the kernel variable `dump_all_pages` is set to one, all memory pages are written to disk, even if they are apparently unused. If the variable is set to zero (with `sysctl`), only kernel pages are saved.

- If set to reboot on panic, the system automatically reboots. Otherwise, the system will wait to be booted manually.

The IRIX, set the kernel variable `reboot_on_panic` to 1 or the PROM variable (with `nvr`) `reboot` to 'y' to reboot automatically.

- After the system reboots, the `savecore` program runs (subject to options in `/var/config`) and performs several tasks:

(1) copies the panic from `swap` to `/var/adm/crash` as long as disk free space is greater than `/var/adm/crash/minfree`, (2) runs `icrash` to produce an analysis

file in `/var/adm/crash`, and (3) reports the panic message to the SYSLOG service.

- *What if the system hangs?* Try to force a panic, so that the system state can be analyzed.
 - If the system has a hardware front-panel interface¹, use the front-panel menus to generate a *Non-Maskable Interrupt* (NMI)². If this does not cause the system to crash, try again. If the second try fails, power down, and power up in power-on-diagnostic (POD) mode — the hardware registers may have information relevant to the problem.

System autopsies

Analyzing a panic is like performing an autopsy, we cannot bring the system back to life, but may find out why it died. In most cases, we won't be able to deduce the cause with certainty — a panic provides only circumstantial evidence. Look for the same pattern in repeated panics.

¹ On a system without a front-panel interface, you can generate an NMI with extensive preparation by (1) installing the debugging kernels (use `inst` to load the `oe.sw.kdebug` package), (2) changing the line `"EXCLUDE: idbg"` in `/var/sysgen/master.d/irix.sm` to `"INCLUDE: idbg"`, and uncommenting the special LDOPTS line, (3) setting the prom variable `console` to `"d"`, (4) install a terminal on `ttyd1`, (5) make sure that `symmon` is in the volume header of the boot disk (use `dvhtool`) (5) reboot, (6) on the terminal type `<CTRL-A>`, then type `"c"` at the DBG prompt.

Now, wait for a hang. When it occurs, type `<CTRL-A>` on the terminal, then type `"call dumpsys"` at the DBG prompt. Voila, a panic. Wasn't that easy?

²On ORIGIN systems, if the front panel does not successfully force a panic, then you can force a panic by plugging a terminal (or laptop) into the MSC control port on the bottom right front corner. After connecting to the controller, type `"<CTRL-T>","NMI <RETURN>","<CTRL-T>","NMI <RETURN>"`. If the system is single ended then type `"FRU1 <RETURN>"`, otherwise type `"FRU2 <RETURN>"`.

On Onyx systems the graphics pipe may panic while leaving Unix running. If graphics locks up, run `/usr/gfx/test/ogtst/tools/kpm -dump`: if the error is in the first part of the pipe ("GE") then you need to make sure at least 256MB of memory is allocated to each graphic pipe. If the error is elsewhere, there is probably a problem with the graphics hardware.

Search for unique error strings, implicated programs (etc.) in the patch database (or in kernel source code, using `Oasis`) Look for incompatibilities among versions of the kernel, add on products, device drivers and hardware. Remember that hardware panics (e.g. memory faults) can be caused by faulty power, connections, board seating, and operating environment (e.g. heat), as well as by particular pieces of hardware. Also remember that panics caused by "running out of resources" can be caused by an application or device driver in a deadlock or infinite loop on resources.

Use `icrash` to analyze the panic. The `icrash` command has many commands, built-in help, the ability to write output to a file, and a built-in `grep` function. The following commands are the most commonly useful ³:

- `report`: generates a report (like that produced by `icrash -r`), basically an abbreviated `stat`, `curproc`, and `trace`, with a `fru` added on high-end systems.
- `stat`: Shows system status, look for:
 - system uptime: if the system uptime is very small, the panic probably occurred on boot-up
 - system version: useful for searching the patch database
 - panic string: this identifies the CPU and the error code. This may be enough by itself to suggest a cause for the panic (e.g. SIMM error) or completely generic
 - `putbuf`: shows the console messages prior to the panic. This may have nothing to do with the panic, but if it fits the general pattern, then treat it as circumstantial evidence.

³The `icrash` command can also be run automatically with the `-r` option. This is roughly equivalent to a slightly abbreviated combination of the `stat`, `curproc`, and `trace` options.

- `curproc (6.3)/curkthread (6.4)`: Shows the running processes at the time of the crash. Think about what system resources the process accesses.

If `curproc` shows "0x0" as the process entry on the panicked CPU, then the panic occurred during an interrupt handler. Under 6.4, the `trace` command will show you a trace of the handler.

- `user -f`: use it to examine the current process to find its arguments⁴ and open files
- `hinv`: gives the hardware inventory of the system at time of panic (use `config` and `memory` under 6.4)
- `files [-p proc] [-n]`: can be used to show open files for a process. Not very useful by itself, but you can sometimes find the name of the file by (1) using `file -n` to get the vnode address (2) using `vnode` with the vnode address to get the vnode data (3) using `inode` on the vnode data address to get the inode number, (4) after rebooting search the original file system with the Unix `find -inum` command.
- `proc`: Shows the process table at crash time. Excessive zombie processes, large groups of processes waiting on the address, or large number of processes with the same parent are all potentially suspicious.
- `trace`: Shows the system call stack at the time of panic. Suspicious system calls are in the middle of the trace — after the generic kernel entry points have been passed but before the PANIC activity itself. Look for unusual system calls and at the subsystem from the system call comes (e.g. an xfs system call could indicate file corruption).
- `nm/sym/symbol`: shows the address of kernel variables. Useful for looking at kernel variable settings. Use `dump` or `print` to find the values associated with each address.

⁴IRIX 6.4 does not have the `user -f` command, instead (1) get the process address with `curkthread`, (2) "`print ((struct proc*) 0xADDRESS)->ps_ui->u_psargs`".

- `print`: shows the content of data structures. Use this to get more information about data structures than the built-in commands give. For example: `print (* (struct *) vnode 0xADDR)` on the address given by the `vnode` command will give *all* vnode information. The include files in `/usr/include/sys` (especially `user.h`, `proc.h`, `vnode.h`) describe kernel data structures.
- `fru`: looks at the hardware error state bits (sbe) and makes a guess about whether the problem was caused by a bad board. Unreliable.