The History and Future of Core Dumps in FreeBSD

Sam Gwydir

December 22, 2016

1 Introduction

Crash dumps, also known as core dumps, have been a part of BSD since it's beginning. A core dump is "a copy of memory that is saved on secondary storage by the kernel"^[1] for debugging a system failure. Though 36 years have passed since `doadump` came about in 3BSD, core dumps are still created and utilized in much the same way they were then. In addition a call to action will be made for modularizing the core dump code.

2 Background

The BSD core dump facility performs a simple yet vital service to the operator: preserving a copy of the contents of system memory at the time of a fatal error, or `panic(9)`. This copy can represent the full binary contents of system memory, a subset of kernel pages that are active at the time of the crash , or a less complete but human readable form referred to as a "textdump".

The FreeBSD variant of the BSD operating system has introduced gradual extensions to this facility in the form of, "minidumps" that only represent kernel memory, "textdumps" consisting of the result of debugger scripting at the time of panic, encrypted dumps, compressed dumps and the ability to dump to a remote network device. While promising, these extensions have been inconsistent in their integration and interoperability.

This paper will provide a historical survey of these dump extensions and describe an active effort to fully modularize them, allowing the operator to enable one or more of them simultaneously. It will also address related utilities to determine the size of a dump in advance and kernel debugger (DDB) scripting options.

3 Motivation

For decades, increases in physical system memory and swap partition size have loosely tracked increases in available persistent memory, allowing for the continued use of the original BSD core dump facilities. Since 4.1BSD, an operator would allocate a region on disk to a "dumpdev" that is equal to physical system memory plus a small buffer.

However, recent advances in commodity system hardware have upended the traditional RAM to disk space ratio with systems now being capable of supporting 2TB or more physical memory running on relatively small solid state disks. Given that the kernel memory footprint has grown in size, the assumption that disk space would always allow for a swap partition large enough for a core dump has proved to be inaccurate. This change in paradigm has spurred development of several extensions to the core dumping facility, including compressed dumping and dumping over the network to a server with disk space for modern core dumps. Network Dumping, or `netdump` does have some security implications which recent work on encrypted dumping may resolve.

4 Quick History

The following is a quick history of core dumps in the BSD operating systems tracing from before the advent of `doadump` in 3BSD through to the present and a discussion of current work on compressed dumps, dumping over the network and encrypted dumps.

Core dumping was initially a manual process as documented in Version 6 AT&T UNIX's `crash(8)`, an operator, "if [they felt] up to debugging" would:

> With a tape mounted and ready, stop the machine, load address 44, and start.

Providing the operator with a core dump on tape to debug a crashed system.

As of 3BSD and with the advent of the LSI-11 core dumping has been automated via `doadump` ^[2], the same function name used today. `doadump` was added to 3BSD in 1980 by Ozalp Babaoglu and was written in 33 lines of PDP-11 assembly.

Beginning in 4.1BSD `doadump` was re-written in C for the VAX and placed in `sys/vax/vax/machdep.c`.

5 References

[1] The Design and Implementation of the FreeBSD operating system by McKusick, Neville-Neil, and Watson [2] crash(8) - 3BSD [3] man 9 panic - `https://www.freebsd.org/cgi/man.cgi?query=panic&apropos=0&sektion=0&manpath=FreeBSD+10.3-RELEASE+and+Ports&arch=default&format=html` [4] `kern_shutdown.c sys/kern/kern_shutdown.c`