Kernel Introspection: A proposal#

What's here#

Background#

Some customers have been asking us for special interfaces into the kernel so they can read ever increasing amounts of information of the like currently provided by debug_process_t and debug_thread_t (sys/debug.h). The motivation has been so customers can write high-availability and monitoring system to control their networks. Such a HA control might want to:

1. Identify and limit CPU and memory hogs
2. Identify and limit FD (open file) hogs
3. Identify over users of shared memory objects
4. Identify threads in mutex-deadlocks
5. Identify thread who's memory and cpu behavoir is unusual (based on history)

Although memory and cpu hogs are well-limited by adaptive partitioning, they are not currently easy to identify. So customers writing sophisticated (aka "bondage-and-domination") high-availablity controllers still need to read this sort of information from the kernel; need to read it for all processes; and need to read it in as quickly as possible. These requests are not well served by the current one-request-at-a-time devctls like DCMD_PROC_STATUS and DCMD_PROC_TIDSTATUS(see pidin_proc.c for example usage).

Then someone asked why we didnt use the MMU to map all of kernel memory so apps could read whatever they want. After we picked ourselves off the floor, we did realize that this question represented the essential dilema. QNX is a microkernel operating system. That meas we have as small a foot print as possible and we have an api which forms a very hard barrier between the kernel and the outside world. That architecture (which we think is way-cool) also means that it will always be a struggle to get lots of interesting stats out of the kernel: it's its effectively asking us to enlarge the kernel and make its immune system (api) more porous.

So this wiki records the proposal, brainstorms, and crude whiteboard notes we've scribbled as way to find a way to provide better accesses to interesting kernel data, without causing the kernel to bloat into, ... expletive deleted..., well, some other kind of operating system.

What is it#

An approach to provide enough visiblity into kernel states to allow customers to build hog detection and system health reporting systems.

There are actually three parts to the proposal:

1. Kernel Bulk Transfer: a uniform and faster way of reading debug/stats from the kernel
2. Kernel Stats Notifier: an asynchronous notification mechanism so apps need not poll the kernel constantly with the bulk transfer mechanism

3. Pathname evolution: a way of using the pathname space to name the data entities both bulk transfer and the kernel stats notifier might be able to talk about.

A nice division. Too bad out all of our design meetings so far are a mish-mash of all three ideas.

Contents#
