# Kernel Introspection:Design Meeting 2007-04-25#

## Who#

bstecher, dbailey, adanko, cburgess

# Summary#

## What we did#

- more on pathnames
- more on api for the generic notifier

## new questions#

- does the current notification scheme support cpu hog detection where the criteria is percentage cpu time averaged over some interval
- should we permit negative levels in thresholds (for the generality)
- an efficient means of notifying proccess creation is desireable: can we deliver the new pid to the client in their sigevent or must they poll.
- should we abandon notification thresholds entirely for cpu hog detection, given the difficulty/inefficiency of using thresholds designed for resource levels
- does /proc/allpids/notifier and /proc/anypid/notifier mean notify me on the set of current pids, or on the set of all pids including ones added in the future
- for thresholding /proc/allpids and /proc/anypid are distinct, are they distinct for bulk transfer?
- does a threshold triggers when the level is > its threshold or >= its threshold. (If the answer is >=, then we do need a signed threshold_level field in order to implement the "trigger on any change" functionality.
- /sumpids/ and /eachpid/ may be clearer than /allpids/ and /anypid/

# More on pathnames #

- To the existing proc paths, like /proc/47821/ let's add /proc/47821/info, which when read returns the tagged process and thread structures, as we would return for bulk transfer, but just for that one process.
- lets use /proc/allpids/info to return tagged process and thread structures for all pids/tids in the system. i.e. bulk transfer
- lets use /notifier instead of /info (wherever /info is legal) to address the notifier device for the same information.
    - for example. notification on a single process memory would be /proc/47821/notifier/memory_usage/
    - notification of a threshold applied to each process, individually, would be: /proc/anypid/notifier/memory_usage
    - notification of a threshold on the sum of memory used by all processes would be: /proc/allpids/notifier/memory_usage.

# API for generic notifier#

This is the struct type seen by the user:

```
struct {
   unint64     threshold_level;
   sigevent    users_event;
   uint32      threshold_flags
}

THRESHOLD_FLAG_LEVEL    0x00000001 /* on: level detection, off: edge detection. recommended: off */
THRESHOLD_FLAG_UP       0x00000002 /* rising edge detect, or trigger  above level */
THRESHOLD_FLAG_DOWN     0x00000004 /* falling edge detect, or trigger below level */
THRESHOLD_FLAG_ONESHOT  0x00000008 /* threshold self-deletes on being triggered */
```

This set of flags allows for these kinds of detections:

- trigger on the one alloc that exceeds a level: UP & NOT LEVEL
- tigger on the one dealloc that falls below a level: DOWN & NOT FLAG_LEVEL
- trigger on any change: FLAG_LEVEL & UP with threshold_level set to 0.
- trigger on every alloc/dealloc when above a level: FLAG_LEVEL & UP
- trigger on every alloc/dealloc when below a level: FLAG_LEVEL & DOWN

Using THRESHOLD_FLAG_LEVEL can cause a flood of sigevents, since it will trigger every time that resource's level changes. So THRESHOLD_FLAG_ONESHOT is offered as a throttling mechanism. It makes sure there is only one notification for each triggering event. (The user must re-create the threshold to re-arm.) With ONESHOT, during the time between triggering the threshold and the user recreating it, we will not be delivering sigenvents. This may be desirable to the user as a means of throttling notifications automatically to a level it can handle for those cases where the user is interested only in the final value of the resource. For other cases, using edge-detecting thresholds gives better throttling.

As an alternative throttling strategy, a user can define their sigevent to be a pulse, so repeating notifications, or a level sensitive threshold, will be compressed.