

Re: [take19 1/4] kevent: Core files.

Re: [take19 1/4] kevent: Core files.

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-10/msg05822.html>

- *From:* Eric Dumazet <dada1@xxxxxxxxxxxxxx>
 - *Date:* Tue, 17 Oct 2006 15:52:34 +0200
-

On Tuesday 17 October 2006 15:42, Evgeniy Polyakov wrote:

On Tue, Oct 17, 2006 at 03:19:36PM +0200, Eric Dumazet (dada1@xxxxxxxxxxxxxx)

wrote:

On Tuesday 17 October 2006 12:39, Evgeniy Polyakov wrote:

I can add such notification, but its existense `_is_` the broken design.

After such condition happend, all new events will dissappear (although they are still accessible through usual queue) from mapped buffer.

While writing this I have come to the idea on how to imrove the case of the size of mapped buffer – we can make it with limited size, and when it is full, some bit will be set in the shared area and obviously no new events can be added there, but when user commits some events from that buffer (i.e. says to kernel that appropriate kevents can be freed or requeued according to theirs flags), new ready events from ready queue can be copied into mapped buffer.

It still does not solve (and I do insist that it is broken behaviour) the case when kernel is going to generate infinite number of events for one requested by userspace (as in case of generating new 'data_has_arrived' event when new byte has been received).

Behavior is not broken. It's quite usefull and works 99.9999% of time.

Re: [take19 1/4] kevent: Core files.

I was trying to suggest you but you missed my point.

You dont want to use a bit, but a full sequence counter, 32bits.

A program may handle XXX.XXX handles, but use a 4096 entries ring buffer 'only'.

The user program keeps a local copy of a special word named 'ring_buffer_full_counter'

Each time the kernel cannot queue an event in the ring buffer, it increase the "ring_buffer_was_full_counter" (exported to user app in the mmap view)

When the user application notice the kernel changed "ring_buffer_was_full_counter" it does a full scan of all file handles (preferably using poll() to get all relevant info in one syscall)
:

I.e. to scan the rest of the xxx.xxx events?

```
do {  
  if (read_event_from_mmap()) {handle_event(fd); continue;}  
  /* ring bu
```