

## Re: async i/o question

---

*Source:* <http://linux.derkeiler.com/Newsgroups/comp.os.linux.development.system/2007-11/msg00022.html>

---

- *From:* Jan Kandziora <jjj@xxxxxx>
  - *Date:* Sun, 04 Nov 2007 00:53:40 +0100
- 

anatolik schrieb:

There could be several application working with same kernel module at the same time.

Then you just misunderstood how the kernel works when processing read(), write() etc.: In process context, the kernel isn't like an application, it's like a library.

Second, there are interrupt routines.

Usually a driver has both things implemented. A part which works in process context and a interrupt routine part which does all the work with the hardware, both communicating through command/data queues.

In process context, your kernel driver has to ensure it does not use single global resources, but take them from a pool instead. Each time your device file is being opened by an application, your driver module has to allocate and initialize some memory for your buffers and control variables which belong to the file descriptor of open(). On close(), your driver has to free that resources. Inbetween, all the read(), write() etc. calls the application makes are passing the file descriptor to your driver module, which can look up the data structure from it. Voila, concurrency enabled by design.

If you have some single resource which cannot be shared (e.g. a physical port), you have to *\*lock\** it some way. Usual way is to implement either a lock on the device node itself, so the device node cannot be opened more than *\*once\**, or (better) to implement two command/data buffer queues, which are filled by write() and emptied by an interrupt routine on sending from the first queue and vice versa on receiving from the second queue. The actual port operation is done by the interrupt routine alone.

Kind regards

Re: async i/o question

Jan

.