Operating Systems
Semester 1, 2014
Eike Ritter

The University of Birmingham
6 November, 2014

Assessed Exercise 3

Deadline: Monday 17 November, 4pm

The Task

Write a device driver for a character device which implements a simple way of message passing. The kernel maintains a list of messages. To limit memory usage, we impose a limit of 4k for each message, and also impose a limit of the size of all messages, which is initially 2MB. Your device driver should perform the following operations:

- Creating the device creates an empty list of messages.
- Removing the device deallocates all messages and removes the list of messages.
- Reading from the device returns one message, and removes this message from the kernel list. If the list of messages is empty, the reader returns -EAGAIN.
- Writing to the device stores the message in kernel space and adds it to the list if the message is below the maximum size, and the limit of the size of all messages wouldn't be surpassed with this message. If the message is too big, -EINVAL is returned, and if the limit of the size of all messages was surpassed, -EAGAIN is returned.
- You should also provide an ioctl which sets a new maximum size of all messages. This operation should succeed only if the new maximum is bigger than the old maximum, or if the new maximum is bigger than the size of all messages currently held. The ioctl should return 0 on success and -EINVAL on failure.

You need to ensure that your code deals with multiple attempts at reading and writing at the same time. Your code should minimise the time spent in critical sections. The reader should obtain the message in the same order as they were written.

For testing you should write programs which read and write data to the device driver, and modify the message limit via ioctl.. These programs are not part of the submission, but you will need them to carry out your testing.

Marking Scheme

Please use the canvas for submitting your code. Please submit only the source code for the kernel module. We will compile and run your code on the virtual machine and mark it accordingly. Please in particular note that we will use the compiler option introduced in the lecture and deduct 6 marks immediately if there is any compiler error or warning.

You should submit precisely ONE file via Canvas, in TAR format, containing a top-level directory "assignment3" in which should appear all the files comprising your submission. Do NOT use a complex directory structure below this.

The end result of invoking your Makefile with the "all" option should be precisely ONE kernel module called "charDeviceDriver.ko". The output file MUST be in the top-level directory after your Makefile completes; if they are not, your work cannot be tested.

We will award marks as follows:

- 5 marks for correctly handling the list of messages.
- 5 marks for correctly handling opening, closing, reading and writing to the device.
- 5 marks for correctly handling the concurrency.
- 5 marks for correctly handling the ioctl.

Extra task

For 5 bonus marks you should implement blocking reads and writes: instead of returning -EAGAIN when reading and writing from the device, you should block the reader until a message is available. Similarly, you should block the writer until there is room for the message (in case of the writer). For the available kernel mechanisms to achieve blocking, see the section "Wait queues and Wake events" in the device driver documentation (available via the module webpage).