Assignment3

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 4KB = 4\*1024 bytes for each message and also impose a limit of the  total number of messages stored in the kernel, which is 1000.

Your device driver should perform the following operations:

* When the module is loaded, the device is created. An empty list of messages is created as well.
* Removing the module deallocates all messages, removes the list of messages and removes the device.
* 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 number of all messages stored in the kernel  wouldn't be surpassed with this message. If the message is too big, *-EINVAL* is returned, and if the limit of the number of all messages was surpassed, *-EBUSY* is returned.
* The kernel module which implements this driver must be called *charDeviceDriver.ko*.

You need to ensure that your code deals with multiple attempts at reading and writing at the same time.  Moreover, your code should handle several read and write attempts  concurrently.  Your critical sections should be as short as possible. The reader should obtain the messages in a FIFO (first in first out) manner.

**Marking Criteria**

30% of the total mark for correctly handling the list of messages

30% of the total mark for correctly handling opening, closing, reading, writing to the device.

30% of the total mark for points for correctly handling the concurrency

10% of the total mark for correctly handling the memory leak

**Testing**

Do not modify or add files in the repository that start with *test*, since we use those files for scripts for testing. The test script expects the kernel module at module loading time to add the major number via printk to the kernel log, which can be viewed via dmesg or  via tail -f /var/log/syslog,, as done in the example device driver file (charDeviceDriver.c in the [archive](https://canvas.bham.ac.uk/courses/65654/files/13907270/download)).

The scripts used for marking will expect this output as well. A [simple test script](https://canvas.bham.ac.uk/courses/65654/files/13930396/download) is available. It should be run from the directory which contains your kernel module.

You should put all your files into a .zip directory named exercise3. The directory exercise3 must be at the top level of your project for the marking scripts to work. So it should have the following structure:

exercise3.zip  
  |\_exercise3  
    |\_charDeviceDriver.c  
    |\_charDeviceDriver.h

We will run the command make in the directory exercise3 in order to obtain all required binaries. Do not use any files or directory with the prefix "test", since we will use such files for running tests.

For marking we will use additional, more advanced, test scripts which check whether your program satisfies the specification. If the provided test script fails, all the more advanced test scripts are likely to fail as well. Any code which does not compile on the virtual machine provided will be awarded 0 marks and not be reviewed.

**Useful commands for this assignment:**  
1. "ls -l /dev/" check the registered device name  
2. "sudo insmod charDeviceDriver.ko" insert the module to kernel  
3. "sudo rmmod charDeviceDriver.ko" remove the kernel module  
4. "dmesg -w &" or "tail -f /var/log/syslog" check the kernel log printed by "printk()"  
5. "mknod /dev/chardev c <major> <minor>" make a device to talk to (check syslog for this, do this before you read or write to device )  
6. "echo <message> > /dev/<devicename>" send message to the device (use "sudo su" change to root user before execute this)  
7. "head -n 1 < /dev/<devicename>" read one message from the device(use "sudo su" change to root user before execute this)