Operating Systems
Assignment 3 – *Hard*

Instructions:

1. The assignment has to be done individually.
2. You can use Piazza for any queries related to the assignment and avoid asking queries on the last day.

1 Character Device Driver

Device drivers are built as loadable kernel modules (LKMs), which can be added to or removed from the kernel at runtime. To uniquely identify a device, Linux assigns *major* and *minor* numbers to a device. The *major* number represents the device type and the *minor* number identifies the device. A device driver receives system calls from users and then appropriately manages the device (internally represented as a file). Hence, the device driver must implement the system calls specific to files: open, close, read, write, lseek, mmap, etc. These operations are described in the fields of the struct *file_operations* structure.

1.1 Problem Statement

Design and implement a character device driver for a virtual LIFO (last in, first out) device (see link), whose size is practically unbounded (can be quite large, let's say limited to 1 MB). We will give it smaller inputs as test cases. You must create two virtual LIFO devices that use the same device driver. One of the devices must be read-only, while the other must be write-only. The characters written to the write-only device can be read only from the read-only LIFO device. The first character read from the read-only LIFO device corresponds to the last character written to the write-only LIFO device. Reading from an empty LIFO device should return an EOF. Otherwise, the process calling the read operation must be blocked, and the process must be woken up once characters are available to be read.

The pseudocode for the reader is shown below:

```
int main(int argc, char *argv[])
{
    char device[MAX_BUF_SIZE];
    char user_msg[MAX_BUF_SIZE];

    strcpy(device, argv[1]);

    fd = open(device, O_RDONLY);

    memset(user_msg, 0, sizeof(user_msg));
    ret = read(fd, user_msg, MAX_BUF_SIZE);
    close (fd);
}
```

The pseudocode for the writer is shown below:

```
int main(int argc, char *argv[])
{
    char device[MAX_BUF_SIZE];
    char user_msg[MAX_BUF_SIZE];

    strcpy(device, argv[1]);
    strcpy(user_msg, argv[2]);

    fd = open(device, O_WRONLY);

    memset(user_msg, 0, sizeof(user_msg));
    ret = write(fd, user_msg, strlen(user_msg));
    close (fd);
}
```

2 Deliverable

1. The character device driver for a virtual LIFO device.
2. Test script that demonstrates the functionality of the driver.
3. Prepare a PDF file with the name A3_report.pdf that includes a description of the driver's functionality, a list of any issues encountered during development and testing.

3 Bonus

The bonus marks will be awarded based on the resilience of the device driver.

4 Submission Instruction

1. We will run MOSS on the submissions. Any cheating will result in a zero in the assignment, a penalty as per the course policy and possibly much stricter penalties (including a fail grade).

2. There will be a demo for the assignment in which you must demonstrate the functionality of the character device driver. This will be followed by a viva in which your general theoretical and practical understanding will be tested (in the context of the assignment). Note that regardless of the code that you submit, the viva performance is vitally important.

3. Create a zip file that contains the report, test script, and a folder that contains the character device driver files, and then name the zip file as, $assignment3\_hard\_<entryNumber>.zip$. Submit this zip file to Moodle. Entry number format: 2020CSZ2445. *Note that all English letters are in capitals.*