EEL 4930/5934 Advanced Systems Programming

Assignment 4

due Friday, March 11th by midnight.

In this assignment you are going to write a character device driver by extending the simple driver we studied in class (see slides DDIntro.pptx) in the following ways:

1. Define a device structure and embed `structure cdev` in that structure:

   ```
   struct asp_mycdrv {
      struct list_head list;
      struct cdev dev;
      char *ramdisk;
      struct semaphore sem;
      int devNo;
   };
   ```

2. Support a variable number of devices that can be set at load time (default will be 3) (see DDScullIntro.pptx). The device nodes will be named /dev/mycdrv0, /dev/mycdrv1, ..., /dev/mycdrvN-1, where *N* is the number of devices. Please have the device driver create the device nodes.

3. **Provide an entry function that would be accessed via `lseek()` function.** That entry function should update the file position pointer based on the offset requested. You should check for bound checks and out of bound requests should be set to the closest boundary.

4. **Provide an entry function that would be accessed via `ioctl()` function.** You should let the user application change the direction of data access: `regular` (default) and `reverse`. `regular` direction means when data needs to be read or written to starting at an offset `o` for `count` number of bytes, the device area, i.e., ramdisk, that is used is `[ramdisk + o, ramdisk + o + count]` (assuming these bounds are legal) so that the first byte in the user buffer, e.g., `buffer[0]`, corresponds to *(`ramdisk + o`), `buffer[1]` corresponds to *(`ramdisk + o + 1`), etc. When `reverse` direction is set, however, the device area that is used is `ramdisk + o - count, ramdisk + o]` (assuming these bounds are legal) and `buffer[0]` corresponds to *(`ramdisk + o`), `buffer[1]` corresponds to *(`ramdisk +`

`o - 1`), etc. You should define symbol `ASP_CHGACCDIR` for the command to change the direction of access and the parameter should be 0 for `regular` and 1 for `reverse`. Your function should set the access mode to requested mode and return the previous access mode. Please see DDScullBasic.pptx for implementation details.

Your read and write functions should support both `regular` and `reverse` access modes and update the file position pointers accurately.

5. **Each device can be opened concurrently and therefore can be accessed for read, write, lseek, and ioctl concurrently.** It is your responsibility to provide appropriate synchronization to prevent race conditions.

6. All the resources (including the ramdisk, the device structures, and device nodes) should be recycled/freed at unloading time.

The assignment is due Friday, March 11th by midnight. Please submit all your files along with a Makefile and a README file on CANVAS.