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 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

In this assignment, we will design and implement a smart block device driver for a loopback-based block device with 4096 sectors, each 512 bytes in size. This driver will introduce additional features beyond a traditional block device driver, focusing on performance optimization, priority management, and dynamic flexibility. The smart block device driver will incorporate the following key features:

1. ***Priority-Aware Asynchronous Write Operations:*** The driver will support non-blocking writes, ensuring that write operations from higher-priority processes are executed before those from lower-priority processes.
2. ***Write Request Caching:*** Write requests will be temporarily stored in an in-memory cache, with the cache size (*in bytes*) specified by the user at the module load time. The write requests will then be served on the device based on the request's priorty.
3. ***Process-Aware Force Flush Mechanism:*** The driver will provide a manual trigger via procfs to flush cached write requests for a specific process. Users can force a flush of all pending write requests for a given process ID (PID) by writing to a proc file.

4. ***Driver Statistics:*** The driver will expose key statistics through procfs, including the current amount of data in the cache and the number of times write requests have been written to the disk.

5. ***Dynamic Cache Size Adjustment:*** Users will be able to dynamically adjust the cache size at runtime without unloading the module. This enhances flexibility in memory management.

2 Deliverable

1. The smart block device driver for a disk.

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 and a list of any issues encountered during development and testing.

3 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 block device driver. A viva will follow, testing your general theoretical and practical understanding within the context of the assignment. Note that the viva performance is vitally important regardless of the code you submit.

3. Create a zip file that contains the report, test script, and a folder that contains the 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.*