# ENSC 351

Fall 2018

## **Final project**

## System calls and kernel drivers

Deadline: End of term
Last revision: November 22, 2018

Karol Swietlicki
Simon Fraser University

## 1 The task at hand

This final project will come in three parts, approximately one per week. All parts will be due at the same point in time, at the end of term.

## 2 Part A: User-mode Linux system call

Create a system call which takes one input, a string.

Print into the kernel log the count of characters in that string.

The use of User-mode Linux is suggested, not required. Anything that gets you working well with the kernel is good enough.

The deliverables for this part are:

1. The file which contains your system call.
2. The system call table file.

## 3 Part B: Simple peek/poke driver

Add to the kernel the support for issuing arbitrary memory reads and writes from userspace.

Create three files inside your /dev/ directory. I would suggest using three separate character device drivers, but if you can make things work otherwise you are welcome to do what you wish.

You will interface with the kernel using reads and writes to those files.

All communication is done in units of fixed size. The exact sizes vary for each of the three files you will be working with.

If you don't have enough bytes arriving in the kernel to complete the task, you need to hold off on performing memory operations and wait for the remaining bytes to make their way into the kernel.

### 3.1 getptr

Implement reads.

Upon read you should emit an 8-byte string which indicates a location of an initially-empty memory area at least 1024 bytes long.

Reads of less or more than 8 bytes are undefined behavior.

### 3.2 peek

Implement both reads and writes.

A complete write is 8 bytes long.

A read is allowed to only get a single byte.

A complete write represents a pointer to a memory location. Any reads after a complete write to the peek file will retrieve a byte from that memory location and send it to the user.

Don't validate your pointer, just do the operation.

### 3.3 poke

Implement writes.

A complete write has 9 bytes.

The first 8 bytes of a complete write represents a memory location. The last byte is data.

Upon a complete write you should use the first 8 bytes as a pointer and write the data byte to the memory address given to you by the pointer.

Don't validate the pointer, just do the operation.

## 4 Part C: Interfacing with the Zedboard hardware

TBD