CS2106: Operating Systems
Lab 5 – Simple File Operations and Mystery

Important:
- The deadline of submission on IVLE is **16th November 5pm**
- The total weightage is 1% + X%:
  - Exercise 1: 1% [**Lab Demo Exercise**]
  - Exercise 2 (and maybe 3): X% [**To be released in week 11**]

Note:
- This lab is **platform sensitive**. Please use only Linux on Intel.

Section 1. Overview

To celebrate the end of lab session for CS2106, this lab should be "a bit lighter" on your time (theoretically). Exercise 1 is a simple task on unix file operations. The lab demo exercise 1 probably takes less than 5 lines of code ☺.

Please note that additional exercise(s) will be released only in week 11 as they requires additional topics covered in lecture 11. **Note that there is no change to the lab demo schedule (i.e. demo only week 11 and 12 as per normal).** There is no formal lab session on week 13, so you can use the lab as needed to finish the additional exercises.

Section 2. Exercise One [Lab Demo Exercise]

Take some time to familiarize yourself with the following Unix file related system calls:

- **`read()`**: Reading data from an opened file descriptor.
- **`lseek()`**: Move to a specified location in the file

You can read through the lecture slide and/or the relevant man pages on Unix for the above functions. Hint: You do not need any other file-related system calls to solve exercise 1 and 2.

Take a look at the sample execution session below. User input in **bold** font.

Sample Run 1	

`File Name: 10int.dat`
`Size = 40 bytes`
`123`
`124`
	One of the provided input data file.
	File size is printed
	Data in file are read and printed (a total of 10 integers in this case).

125 126 127 128 129 130 131 132	

Sample Run 2	
File Name: **wrong.dat** Cannot Open	This is a non-existent file. Complain and exit.

The given skeleton file **ex1.c** has quite a large chunk of logic implemented. Your tasks are essentially:

1. Check that the file can be opened.
2. Find out the file size (hint: use **lseek()**).
3. Read all data until end of the file.

A note on 32-bit vs 64-bit

The exercises in this lab are sensitive to the word size of the underlying execution environment. As some of you may have a 64-bit processor and OS, it is a little tricky to ensure the lab works across both 32-bit and 64-bit environments. For maximum compatibility, we have decided to stick to 32-bit for this lab.

To ensure the correct execution environment, the first part of the skeleton source will do a simple check on the integer size and warn you if your machine + compiler operates in 64-bit. It will terminate the program if 64-bit environment is detected.

The remedy is quite simple, you just need to compile your source code with an additional flag "**-m32**", e.g.

gcc ex1.c -m32 //compiles as 32-bit application.

Please make sure you compile your code correctly.

For your own exploration:

1. If you open up the **10int.dat** in a normal editor, what do you see? Can you explain?