Due: Midnight, March 5, 2020

## 1. `pthreads` API

Write a "C" program that will:

a) Accept two parameters from the command line:
   i. The number of threads to start
   ii. Length of the string to generate
b) Program will start up the number of threads specified. A thread number (integer) should be passed to each thread to identify its output.
c) Each thread will add characters (of your choice) to a string until it reaches the length specified. Make sure this operation is atomic! When a thread discovers that the string is already of the required length, it will terminate. Thread should print out each character (and the thread number) added.
d) After all threads have terminated, print the final string.

Notes:

i. Strings in "C" must be terminated by NULL (0) characters. A simple way to handle this is for the main to initialize the memory to all nulls (lookup the `memset()` function) .
ii. There are a number of ways the string memory could be acquired and the address passed to the thread function. One simple approach would be to have a global pointer and `malloc()` the required size (remember space for a trailing NULL). Another approach would be to allocate the string storage on the stack and pass its address to the thread function.

## 2. Locking

A locking simulator (locks.zip) has been uploaded to Oaks. The zip archive contains several files:

- x86.py – The simulator program
- README.txt – A description of the simulator
- flags.s, test_and_set.s and yield.s – Simple "assembler" programs

As with the previously python programs downloaded, you will run the program using Python version 2 installed in the Linux virtual machine.

a) Read the README file to get an idea how the simulator works.
b) Run the simulator on each of the "assembler" programs with the following parameters specified:

i. `-p program.s` - Name of the .s file to run
ii. `-c` - Compute results
iii. `-a bx=10` - Set the bx register to10 (number of loop iterations)
iv. `-M count` - Display the value of the count variable
v. `-i 5` - Interrupt every 5 instructions
vi. `-t 3` - Run three tasks
vii. `-S` - Print extra statistics

c) The final count in each case should be 30. For each of the programs:
    i. If this is not true, explain why it isn't correct.
    ii. If it is true explain why it is correct.
d) What is the total number of instructions executed for each program. Why do you think it varies as much as it does?
e) Run the programs again varying the -I parameter. For each program explain:
    i. What happens to the count value?
    ii. Does the number of instructions executed change?