Due: Midnight, February 6, 2020

1. Fork and Exec

Write a "C" program that will:

a. Accept a command (with parameters) as a parameter
b. Fork creating a new process
c. Execute the command passed as the parameter printing the results

For example, if your program is named hw3 and you ran, from the command line:

```
./hw3 ls -l
```

Your program would fork, then execute the 'ls -l' command and display its output (a listing of the current directory).

Submit your source code and a make file to build the program.

2. Simple Scheduling

a) FIFO Scheduling

Using a First In, First Out (FIFO) scheduler and given the following set of jobs, compute and display:

- The start and end times for each job
- The turnaround time for each job
- The response time for each job
- The average turnaround and response times for the set of jobs

Jobname	**Execution Time (Seconds)**
Job 1	10
Job 2	15
Job 3	5
Job 4	22

b) SJF Scheduling

Using a Shortest Job First (SJF) scheduler and given the following set of jobs, compute and display:

- The start and end times for each job
- The turnaround time for each job
- The response time for each job
- The average turnaround and response times for the set of jobs

Jobname	Execution Time (Seconds)
Job 1	25
Job 2	15
Job 3	5
Job 4	10

c) STCF Scheduling

Using a Shortest Time to Completion First (STCF) scheduler and given the following set of jobs, compute and display:

- The start and end times for each job
- The turnaround time for each job
- The response time for each job
- The average turnaround and response times for the set of jobs

Jobname	Arrival Time	Execution Time (Seconds)
Job 1	0	25
Job 2	0	15
Job 3	5	5
Job 4	10	10

d) RR Scheduling

Using a Round Robin (RR) scheduler and given the following set of jobs, compute and display:

- The start and end times for each job
- The turnaround time for each job
- The response time for each job
- The average turnaround and response times for the set of jobs

Jobname	Arrival Time	Execution Time (Seconds)
Job 1	0	20
Job 2	0	10
Job 3	5	5
Job 4	10	15

3. More Complex Scheduling

For each of these problems you will need to run a scheduling simulator. The simulators are available in Oaks (Content->Homeworks->Homework-3) and each includes a README file describing their use.

To run the simulators, you will need Python 2 (NOT python 3). In your Linux VM, Python 2 may be installed by default (run `python -V` to tell). It it's not installed, you can install it (assuming you're using the osboxes.org VM) with the command:

```
sudo apt install python.
```

With python 2 installed, you can run the schedulers (in this example the mlfq) using, the command:

```
python mlfq.py other-parameters
```

a) MLFQ Scheduling

Run the MLFQ scheduler (`mlfq.py`) using the following parameters:

Number of jobs (-j):	4
Random Seed (-s):	13
Compute Results(-c)	

i) What are the turnaround and response times?

Run the MLFQ scheduler again with the same parameters but vary the priority boost time (-B).

ii) What happens to the turnaround and response times as you increase the boost time?

b) Lottery Scheduling

Run the Lottery scheduler (`lottery.py`) using the following parameters:

Number of jobs (-j):	4
Random Seed (-s):	17
Compute Results(-c)	

i) What are the turnaround and response times? You will need to work through the output to compute the times.

Run the lottery scheduler again with the same parameters but vary the time slice time (-q).

ii) What happens to the turnaround and response times as you increase the boost time?