Operating Systems Assignment 2 (CLO 1, PLO1, Total Marks 40)

This program, process-run.py, allows you to see how process states change as programs run and either use the CPU (e.g., perform an add instruction) or do I/O (e.g., send a request to a disk and wait for it to complete).

1. [4] Run process-run.py with the following flags: -l 8:100,2:100. What should the CPU utilization be (e.g., the percent of time the CPU is in use?) Why do you know this? Use the -c and -p flags to see if you were right.

2. [4] Now run with these flags: ./process-run.py -l 6:100,1:0. These flags specify one process with 6 instructions (all to use the CPU), and one that simply issues an I/O and waits for it to be done. How long does it take to complete both processes? Use -c and -p to find out if you were right.

3. [4] Switch the order of the processes: -l 1:0,6:100. What happens now? Does switching the order matter? Why? (As always, use -c and -p to see if you were right)

4. [4] We'll now explore some of the other flags. One important flag is -S, which determines how the system reacts when a process issues an I/O. With the flag set to SWITCH ON END, the system will NOT switch to another process while one is doing I/O, instead waiting until the process is completely finished. What happens when you run the following two processes (-l 2:0,8:100 -c -S SWITCH ON END), one doing I/O and the other doing CPU work?

5. [4] Now, run the same processes, but with the switching behavior set to switch to another process whenever one is WAITING for I/O (-l 2:0,8:100 -c -S SWITCH ON IO). What happens now? Use -c and -p to confirm that you are right.

6. [4] One other important behavior is what to do when an I/O completes. With -I IO RUN LATER, when an I/O completes, the process that issued it is not necessarily run right away; rather, whatever was running at the time keeps running. What happens when you run this combination of processes? (Run ./process-run.py -l 4:0,4:100,4:100,4:100 -S SWITCH ON IO -I IO RUN LATER -c -p) Are system resources being effectively utilized?

7. [4] Now run the same processes, but with -I IO RUN IMMEDIATE set, which immediately runs the process that issued the I/O. How does this behavior differ? Why might running a process that just completed an I/O again be a good idea?

8. [12] Now run with some randomly generated processes: -s 1 -l 4:50,4:50 or -s 2 -l 4:50,4:50 or -s 3 -l 4:50,4:50. See if you can predict how the trace will turn out. What happens when you use the flag -I IO RUN IMMEDIATE vs. -I IO RUN LATER? What happens when you use -S SWITCH ON IO vs. -S SWITCH ON END?