1processes-lab

CST 334 (Operating Systems)
Dr. Glenn Bruns

Lab: Processes

Try to answer problems 1-4 without referring to the lecture slides or other sources, and write down your answer.

1. Define the term 'process'.

2. If you want to stop a process and start it again later, what information do you need to store about the process?

3. Define the term "multiprogramming".

4. What is stored in a program's heap?

5. Log into mlc104, and Enter `ps -ef`. It will show you all processes running on the hosting machine. Do you see processes owned by people you know? Are all the users regular human users?

6. Look at the man page for 'ps'. Skim the page, looking at the summary and some of the options.

7. Run command `top`. What does it show you? Is any process using a lot of CPU?

8. Next we will use the simulator provided by the authors of our text.

Copy these files to your home directory on mlc104:

- `/home/CLASSES/brunsglenn/OSTEP/HW-CPU-Intro/process-run.py`
- `/home/CLASSES/brunsglenn/OSTEP/HW-CPU-Intro/README-process-run`

They are also available on the web at:

http://pages.cs.wisc.edu/~remzi/OSTEP/Homework/homework.html

Your job will be to simulate the execution of some processes, and to show they state they are in during each step of execution.

Read the top of the README file. It explains that, in the simulation, processes can be in one of four states: RUNNING, READY, WAITING, or DONE. The difference between READY and WAITING is that in the READY state a process could run immediately if it had the CPU, while in the WAITING state a process needs to wait for I/O to finish. (The WAITING state is called the "blocked" state in Chapter 4.)

9. Run the program as follows from your home directory:

   - `./process-run.py -h`

You should see all the command options for the program.

10. Run the program as follows:

- `./process-run.py -l 3:100,3:100 -c`

The `-l` flag is used to give a list of processes. In this case, two processes are specified: one of three instructions, where the instructions are 100% CPU, and another of three instructions, where the instructions are also 100% CPU (in other words, no I/O instructions such as disk reads).

The `-c` flag shows that the schedule will be computed by the program and displayed.

11. Run the program as follows:

- `./process-run.py -l 2:0,5:100 -c`

Look at the process list. It says: run two processes, the first with 2 I/O instructions, the second with 5 CPU instructions. Look carefully at the processes and how their states change. Can you explain the scheduling? How long does I/O take to complete? Don't rush this problem.

Also, run the program this way, adding -p at the end:

- `./process-run.py -l 2:0,5:100 -c -p`

Explain the statistics that are now produced by the simulation.

12. Now, answer question 1 at the end of Chapter 4. Write down your prediction of how the processes will be scheduled. After you've done this, run the command again, adding the -c flag.

13. Answer question 2 at the end of Chapter 4

14. Answer question 3 at the end of Chapter 4.

15. If you still have time, continue and answer questions 4-8 at the end of Chapter 4.