OS Lab 2

Goal:

The goal of this lab is to give you a basic introduction to the OS, shell, and processes in general. You will run a few simple programs (as processes on your Linux machine), and monitor their behavior using the proc file system and other related tools.

Before you start:

• Understand the proc file system of linux. A web search or man proc should give you a lot of information. In particular, understand the stat and status files of a process, and the system wide files of the proc directory.

• Learn how to use the following basic commands. top tells you information about processes and their resource consumption. ps tells you information about all processes in the system. ps has several useful commandline arguments. It is worthwhile understanding and learning the most useful ones. For example, ps -A gives you information about all processes in the system. Also, learn to use any disk monitoring tool. For example, iostat gives you information about disk utilization. Again, understanding various useful commandline arguments (-c -x -m etc.) will be quite handy.

Exercise:

Do the following exercises, and record your observations in your report.

1. Collect the following basic information about your machine using proc. How many CPU cores does the machine have? How much memory, and what fraction of it is free? How many context switches has the system performed since bootup? How many processes has it forked since bootup?
2. Recall that every process runs in one of two modes at any time: user mode and kernel mode. It runs in user mode when it is executing instructions / code from the user. It executes in kernel mode when running code corresponding to system calls etc. Compare (qualitatively) the programs cpu and cpu-print in terms of the amount of time each spends in the user mode and kernel mode, using information from the proc file system. For examples, which programs spend more time in kernel mode than in user mode, and vice versa? Read through their code and justify your observations.
3. Open a bash shell. Find its pid. Write down the process tree starting from the first init process (pid = 1) to your bash shell, and describe how you obtained it. You may want to use the pstree command.
4. Consider the following commands that you can type in the bash shell: cd, ls, history, ps. Which of these are system programs that are simply executed by the bash shell, and which are implemented by the bash code itself?
5. Run the following command in bash.
   $./cpu-print > /tmp/tmp.txt &

   Find out the pid of the new process spawned to run this command. Go to the proc folder of this process, and describe where its I/O file descriptors 0, 1, 2 are pointing to. Can you describe how I/O redirection is being implemented by bash?

8. Run the following command with cpu-print.
$./cpu-print | grep hello

Once again, identify which processes are spawned by bash, look at the file descriptor information in their proc folders, and use it to explain how pipes work in bash.

Submission and Grading:

You may solve this assignment in groups of at-most two students. You must submit a tar gzipped file, whose filename is a string of the roll numbers of your group members separated by an underscore. For example, rollnumber1_rollnumber2.tgz. The tar file should contain the following:
• report.pdf, which contains your answers to the exercises above. Be clear and concise in your writing.

Evaluation of this lab will be based on reading your answers to the exercises in your report.