Lab Assignment 2

August 19, 2020

Objective

This assignment introduces simple tools available in Linux which helps in understanding some fundamental concepts related to process management and memory management

Acknowledgement

The last assignment has been borrowed from the *Lab : Introduction* slide of the Operating Systems course offered by **Professor Mythili Vutukuru**, Department of Computer Science and Engineering, IIT Bombay.

Thanks to Professor Mythili.

Instructions

- Estimating memory Usage using the *free* tool
  - Login to your Linux systems, open a terminal and type *man free*. Read about the *free* tool.
  - Now, run *free* with *-m* argument. The *-m* argument displays memory in Megabytes. How much memory is in your system? How much is free?
  - Create a program that uses a certain memory. Call it *memory_use.c*. The program should take as a command line argument: the number of Megabytes of memory it will use. When run, it should allocate an array, and constantly stream through the array, touching each entry. The program should do indefinitely, or for some amount of time specified at the command prompt.
  - While running *memory_use.c*, on the same machine but in a different terminal window run the *free* tool. How do the memory usage change when the program is running?
  - What is the memory usage when you terminate (*kill*) the program?
  - Try the above steps for different amounts of memory usage. What happens when large amount of memory is used?

- Process Management tools in Linux
  - Use the *ps, ps lx, ps tree and ps -aux* command to display the process attributes
  - Learn the *top* command to display the resource utilisation statistics of processes. Try other variants of *top* command such as *htop*, *atop* and *vtop*. To install *htop* type on the terminal : *sudo apt-get install htop*
  - * Open a terminal and type the *top* command
    * Start a browser and see the effect on the *top* display
    * Compile a C program and observe the same effect (Use a long loop - say while(1) to observe the effect)
    * From the *top* display, answer the following:
      · How much memory is free in the system?
      · Which process is taking more CPU?
      · Which process has got maximum memory share?
    * Write a CPU bound C program and a I/O bound C program (e.g. using more printf statements within while(1) loop), compile and execute both of them.
      Observe the effect of their CPU share using the *top* display and comment.
  - Working with the *proc* file system
    * 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 about the *iostat* command. *iostat* gives you information about disk utilization.
  - Working with the *pmap* tool
    * Read the manual page of *pmap*
    * To use *pmap*, one needs to know the **process ID** of the process of interest. Thus, run *ps auxw* to list the processes; then pick one of them, say the *browser* . Now, run *pmap* using various flags like **pmap -X pid** to reveal details about the process. What do you see? How many different entities make up the address space? Does it contain only code, stack and heap or something else?
    * Run *pmap* on the *memory_use.c* program with different amounts of used memory. What do you observe?

- Four simple C programs have been provided to you for this lab in the *lab2_related_files* folder :

  cpu.c, cpu-print.c, disk.c and *disk1.c.*

  Compile each and keep all the executables ready for solving the assignments in the forthcoming labs.

Compile the program *cpu.c* given to you and execute it in the bash or any other shell of your choice as follows.

$ gcc cpu.c -o cpu $ ./cpu

This program runs in an in?nite loop without terminating. Now open another terminal, run the top command and answer the following questions about the cpu process.

- What is the PID of the process running the cpu command?
- How much CPU and memory does this process consume?
- What is the current state of the process? For example, is it running or in a blocked state ?

- Consider the two programs memory1.c and memory2.c given to you. Compile and run them one after the other. Both programs allocate a large array in memory. One of them accesses the array and the other doesn?t. Both programs pause before exiting to let you inspect their memory usage. You can inspect the memory used by a process with the ps command. In particular, the output will tell you what the total size of the ?virtual? memory of the process is, and how much of this is actually physically resident in memory. You will learn later that the virtual memory of the process is the memory the process thinks it has, while the OS only allocates a subset of this memory physically in RAM.

- In this question, you will compile and run the programs *disk.c* and *disk1.c* given to you. These programs read a large number of files from disk, and you must first create these files as follows. Create a folder disk-files and place the file *foo.pdf* in that folder. Then use the script *make-copies.sh* to make 5000 copies of the same file in that folder, with different file names. The disk programs will read these files. Now, run the disk programs one after the other. For each program, measure the utilisation of the disk while the program is running. Report and explain your observations. You will find a tool like *iostat* useful for measuring disk utilisation. Also read through the code of the programs to help explain your observations.