Lab 2 : Operation on Process

August 12, 2015

Objective :

- Lab 2 is intended to learn how to create and work with how to manipulate processes in Linux.

Recommended Systems/Software Requirements:

- Any flavour of Linux

References:

1. *Unix concepts and applications*, Fourth Edition, Sumitabha Das, TMH.

Theoretical Background:

You are expected to refer to the text book and references mentioned in the course wensite befor you start the lab.

Assignments:

1. Use the *ps* command to display the process attributes.
2. Learn the *top* command to display the resource utilization statistics of processes
   - 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 withing while(1) loop), compile and execute both of them.
     Observe the effect of their CPU share using the *top* display and comment.

3. Write a program in C that uses the *fork* system call. You can use the program described in the lecture class on **Process**. Compile the program and execute it.

4. For the above C program, change the program such that the associated child process will change the core image.

5. In a C program, print the address of the variable and enter into a long loop (say using while(1)).

   - Start three to four processes of the same program and observe the printed address values.
   - Try the experiment of different OS and comment whetheher the addresses remain same on both OS or not?

6. Use *strace* command to find out system call traces of an executing process. You can use any process that has been crteated earlier.

   - Find out a command on the shell such that the command does not make a system call. Use *strace* to locate such a command.
   - *strace* **bash** to observe how bash uses system calls to read commands from the console and echo it back to screen.

7. Write a C program to create threads. You can use the program demonstrated in class which has been provided as additional resource in the course website.