**INDIAN INSTITUTE OF TECHNOLOGY**

**GOA**

**OPERATING SYSTEMS LAB (377)**

Faculty: Dr. Sharad Sinha

Teaching Assistant: Pavitra P. Bhade

**Lab 02**

In this lab, you will learn to create, work with and manipulate processes in Linux. This lab exercise will be split in two parts.

**PART A:**

In this part, you will be introduced to various system calls and commands in process management. A process refers to a program in execution; it’s a running instance of a program.

There are fundamentally two types of processes in Linux:

* Foreground processes (also referred to as interactive processes) – these are initialized and controlled through a terminal session. In other words, there has to be a user connected to the system to start such processes; they haven’t started automatically as part of the system functions/services.
* Background processes (also referred to as non-interactive/automatic processes) – are processes not connected to a terminal; they don’t expect any user input.

There are two conventional ways used for creating a new process in Linux:

* Using The System() Function – this method is relatively simple, however, it’s inefficient and has significantly certain security risks.
* Using fork() and exec() Function – this technique is a little advanced but offers greater flexibility, speed, together with security.

Because Linux is a multi-user system, meaning different users can be running various programs on the system, each running instance of a program must be identified uniquely by the kernel.

A program is identified by its process ID (PID) as well as it’s parent processes ID (PPID), therefore processes can further be categorized into:

* Parent processes – these are processes that create other processes during run-time.
* Child processes – these processes are created by other processes during run-time.

In order to understand the working of fork, exec, exit and wait system calls, some reference material is provided along with this manual. You are expected to compile and run the same.

The commands like ps, ps lx, pstree and ps -aux are used to study the process attributes. Commands like top, are used to display the resource utilisation statistics of a process.

**PART B:**

Perform the following exercises:

1. Use the ps, ps lx, pstree and ps -aux command to display the process attributes.
2. Learn the top command to display the resource utilization statistics of processes
   1. Open a terminal and type the top command
   2. Start a browser and see the eﬀect on the top display
   3. Compile a C program and observe the same eﬀect (Use a long loop say while(1) to observe the eﬀect)
   4. 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?
   5. 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.
3. Write a program in C that creates a child process, waits for the termination of the child and lists its PID, together with the state in which the process was terminated (in decimal and hexadecimal)
4. In a C program, print the address of the variable and enter into a long loop. Start three to four processes of the same program and observe the printed address values. Show how two processes which are members of the relationship parent child are concurrent from execution point of view, initially the child is copy of the parent, but every process has its own data.