Lab 4
Process

4.1 Process Termination

Processes termination occurs either of two ways

1. *Normal exit:* when the program calls exit function or or program, main function return.
2. *Killed by another process:* Terminated abnormally in response to a signal.
   kill [signal] pid

Signals: (signal system calls are defined under the header <signal.h> and <sys/type.h>)

KILL – Forcefully make free (used in hang).
TERM – Kill process.
e.g. kill -KILL pid -send the signal to the targeted pid process.

4.1.1 Waiting for process termination

Waiting can be done with the *wait* family of system call. These functions allow to wait for process to finish executing, and enable the parent process to retrieve information about child's termination.

4.2 Processes States

The basic processes states in Linux are
Running - R, Sleeping – S, Stopped – T and Zombie – Z.
If the parent process terminated before child process, the executing child is a orphan process.
A zombie process is a process that has terminated but has not been cleaned up yet.
If the child process finish before the parent process calls wait, the child process becomes zombie.

$ ps -el -look process with state information.

Ex 4.1: Checking process state (pstat.c).

```
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>

int main()
{
int pid;
pid = fork();
   if (pid = = 0)
         {
              printf("I am the child, my pid is %d\n", (int) getpid());
              printf("My parent's pid is %d \n", (int) getppid());
              sleep(20);
```

```
                printf("I am the child, my pid is %d\n", (int) getpid());
                printf("My parent's pid is %d \n", (int) getppid());
            }
        else
            {
                sleep(10);
                printf("I am the parent, my pid is %d\n", (int) getpid());
                printf("My parent's pid is %d \n", (int) getppid());
                for(;;);
            }
    }
```

Ex 4.2 : Process switching (pswitch.c)

```
    #include<unistd.h>
    #include<stdio.h>
    int main()
    {
    int pid;
    pid =fork();
    if (pid == 0)
        {
            for(;;)
                printf("C");
        }
    else if (pid > 0)
        {
            for (;;)
                printf("P");
        }
    }
```

Assignment #L4

1. Run the program Ex 4.1 and issue the command *ps -el* three times in every 10 seconds and analyze the output.
2. Modify the program Ex 4.1 to see running, sleeping and zombie state of program individually.
3. Insert the wait system call in the parent portion of program Ex 4.1 and analyze the effect.
4. Run the program Ex 4.2, checks states and analyze the output.