Lab 3 – PROCESSES

TASK 3.4

Using `fork()`: fork command in Linux creates a new process by duplicating the calling process. The new process, referred to as the **child**, is an exact duplicate of the calling process, referred to as the **parent**. What is the outcome of the following program?

```
int main(){
     fork();
     printf("The PID is %d\n", getpid());
     return 0;
}
```

TASK 3.5

What is the outcome of the following program?

```
int main(){
     int pid;
     pid = fork();
     if(pid==0){
          printf("I am child, my process ID is %d\n", getpid());
          printf("The  parent process ID is   %d\n", getppid());
     }
     else{
          printf("I am parent, my process ID is   %d\n", getpid());
          printf("The parent process ID is   %d\n", getppid());
     }
     return 0;
}
```

TASK 3.6

To see if the pid is same as shown in the system, Open System Monitor. Check to see if the pid is same. Use the following code

```
int main(){
     int pid,i;
     pid = fork();
     if(pid==0){
          printf("I am child, my process ID is   %d\n", getpid());
          printf("The  parent  process ID is   %d\n", getppid());
     }
     else{
          printf("I am parent, my process ID is   %d\n", getpid());
          printf("The parent process ID is   %d\n", getppid());
     }
     scanf("%d",&i);   //so that program halts for user input
     return 0;
}
```

Show screenshots here: -

TASK 3.7

What is the outcome of this program?

```
/**
 * This program forks a separate process using the fork()/exec()
system calls.
 * * Figure 3.10*
 * @author Gagne, Galvin, Silberschatz Operating System Concepts -
Seventh Edition
 * Copyright John Wiley & Sons - 2005. */
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
int main(){
pid_t pid;
     /* fork a child process */
     pid = fork();
     if (pid < 0) { /* error occurred */
          fprintf(stderr, "Fork Failed\n");
          exit(-1);
     }
     else if (pid == 0) { /* child process */
          printf("I am the child %d\n",pid);
          execlp("/bin/ls","ls",NULL);
     }
     else { /* parent process */
          /* parent will wait for the child to complete */
          printf("I am the parent %d\n",pid);
          wait(NULL);

          printf("Child Complete\n");
          exit(0);
     }
}
```