Name – Moitrish Maity

Roll – 48 3rd year 5th sem

Lab on Process

1)

```
#include <stdio.h>
#include<sys/types.h>
#include<stdlib.h>
int main(void)
{
   pid_t pid;
   pid = fork();
   if(pid>0)
   {
      sleep(1);
      printf("PARENT PROCESS---PID : %d  CHILD PID :%d\n",getpid(),pid);
   }
   else if(pid==0)
   {
      printf("CHILD PROCESS--PID : %d  PARENT PID : %d\n",getpid(),getppid());
   }
   else
   {
      printf("Fork error!\n");
      exit(1);
   }
```

```
    exit(0);
}
```

2)

```
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>
int main (void)
{
    pid_t pid;
    pid = fork();
    if(pid>0)
    {
        printf("PARENT PROCESS---PID : %d  CHILD PID : %d\n",getpid(),pid);
    }
    else if(pid == 0)
    {
        printf("CHILD PROCESS---PID : %d  PARENT PID : %d\n",getpid(),getppid());
        sleep(4);
        printf("AFTER SLEEP, NEW CHILD PROCESS---PID : %d  NEW PARENT PID : %d\n",getpid(),getppid());
    }
    else
    {
        printf("Fork Error!!");
```

```
    exit(1);
  }
```

3)

```
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
        int pid = fork();
        if (pid > 0)
                sleep(20);
        else
        {
                exit(0);
        }
        return 0;
}
```

Questionnaires

1. What do you understand by a child process?

A child process is a process created by a parent process in operating system using a fork() system call. A child process may also be called a subprocess or a subtask.A child process is created as its parent process's copy and inherits most of its attributes. If a child process has no parent process, it was created directly by the kernel.

2. Discuss the fork() function.

fork() function is used to create processes. It takes no arguments and returns a process ID. The purpose of fork() is to create a new process, which becomes the child process of the caller. After a new child process is created, both processes will execute the next instruction following the fork() system call. Therefore, we have to distinguish the parent from the child. This can be done by testing the returned value of fork().

3. Discuss the significance of pid and ppid of a process.

PID or Process Identifier is a unique number to identify each process running in an operating system such as Linux, Windows, and Unix. PIDs are reused over time and can only identify a process during the lifetime of the process, so it does not identify processes that are no longer running.

PPID or Parent Process Identifier is the PID of the process's parent.

When processes go bad,we might try to quit a program only to find that it has other intentions. The process might continue to run or use up resources even though its interface closed. Sometimes, this leads to what is called a zombie process, a process that is still running, but dead.One effective way to kill a zombie process is to kill its parent process. This involves using the ps command to discover the PPID of the zombie process and then sending a kill signal to the parent. Of course, any other children of the parent process will be killed as well.That is why PID and PPID of a process is significant.

3. When a process is called zombie? How do you demonstrate the existence of zombie process?

A process in Unix or Unix-like operating systems becomes a zombie process when it has completed execution but one or some of its entries are still in the process table. If a process is ended by an "exit" call, all memory associated with it is reallocated to a new process.

If the parent decides not to wait for the child's termination and executes its subsequent task, then at the termination of the child, the exit status is not read. Hence, there remains an entry in the process table even after the termination of the child. This state of the child process is known as the Zombie state.

4. When a process is called orphan? Discuss its difference with a zombie process. How do you demonstrate the existence of orphan process?

A process whose parent process no more exists i.e. either finished or terminated without waiting for its child process to terminate is called an orphan process.

A Zombie is created when a parent process does not use the wait system call after a child dies to read its exit status, and an orphan is child process that is reclaimed by init when the original parent process terminates before the child.Orphan - Parent exit , Init process becomes the parent of child process. Whenever child is terminated, process table gets deleted by os.Zombie - When the child terminates it gives exit status to parent. Meanwhile time suppose your parent is in sleep state and unable to receive any status from child. Though the child exit but the process occupies space in process table.