Lab-1

Introduction to Shell Command Process

Theory:

A shell is a special user program which provides an interface to the user to use operating system services. Shell accept human readable commands from the user and convert them into something which kernel can understand. Some of the basic shell commands are:

- mkdir :- It is used to create a directory.
- ls : It is used to get the list of all the files or folders.
- cd : It is used to change the directory.

Process:

A process is defined as an entity which represents the basic unit of work to be implemented in the system.

fork():

Fork system call use for creates a new process, which is called child process, which runs concurrently with process known as parent process. After a new child process is created, both processes will execute the next instruction following the fork() system call.

getpid():

It returns the process ID of the calling process.

getppid():

It returns the process ID of the parent of the calling process.

Orphan process

An orphan process is a computer process whose parent process has finished or terminated, though it remain running itself.

Zombie process:

A process which has finished the execution but still has entry in the process table to report to its parent process is known as zombie process.

Source code:

Program 1: (lab1.c)

```
#include<stdio.h>
#include<sys/types.h>
int main()
{
    fork();
    fork();
    printf("Hello World\n");
    return 0;
}
```

Input and output:

```
gcc -c lab1.c -o lab1.o
gcc lab1.o -o lab1
./lab1
```

Output:

```
Hello World
Hello World
Hello World
Hello World.
```

Discussion:

Here, fork() function was used which creates a child

Program 2: (lab2.c)

Source code:

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
int main()
{
    int pid;
    pid = fork();
    if (pid > 0)
    {
        printf("Child PID: %d \n Parent PID = %d", getpid(),
                getppid());
    }
    else
    {
        exit(0);
    }
    return 0;
}
```

Input and Output:

Input:
```
gcc -c lab2.c -o lab2.o
gcc lab2.o -o lab2
./lab2
```

Output:
```
Child PID =
Parent PID =
```

Discussion:

In this program, getpid()