RIPHAH INTERNATIONAL UNIVERSITY

Operating System

Lab 8

Submitted By:

Mahnoor Farrukh (52926)

BSCS-5

Task 1:

```c
#include<stdio.h>
#include<unistd.h>



int main (){
pid_t pid1 = fork();

if(pid1 < 0 )
{
perror("Fork 1 failed");
return 1;

}

pid_t pid2 = fork();

if(pid2 < 0 )

{ perror("Fork 2 failed ");
return 1;
}


for(int i=1; 1<=20; i++)
{
  printf("Process ID: %d |Loop Value:%d\n  ", getpid(),i);
```

```
mahnoor@mahnoor-VMware-Virtual-Platform:~$ pico 1.c
mahnoor@mahnoor-VMware-Virtual-Platform:~$ gcc 1.c -o 1
mahnoor@mahnoor-VMware-Virtual-Platform:~$ ./1
Process ID: 3702 |Loop Value:1
Process ID: 3704 |Loop Value:1
Process ID: 3703 |Loop Value:1
Process ID: 3705 |Loop Value:1
  Process ID: 3702 |Loop Value:2
  Process ID: 3704 |Loop Value:2
  Process ID: 3703 |Loop Value:2
  Process ID: 3705 |Loop Value:2
  Process ID: 3704 |Loop Value:3
  Process ID: 3702 |Loop Value:3
  Process ID: 3703 |Loop Value:3
  Process ID: 3705 |Loop Value:3
  Process ID: 3702 |Loop Value:4
  Process ID: 3704 |Loop Value:4
  Process ID: 3703 |Loop Value:4
  Process ID: 3705 |Loop Value:4
  Process ID: 3704 |Loop Value:5
```

Task 2:

```
mahnoor@mahnoor-VMware-Virtual-Platform:~$ nano c2.c
mahnoor@mahnoor-VMware-Virtual-Platform:~$ cat c2.c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>

int main()
{
pid_t pid;
int i;

for(i = 0; i < 3; i++) {

pid = fork();
if(pid==0) {
   printf("Child %d = PID : %d, PPID : %d\n" , i+1, getpid(), getppid());
    exit(0);

}  else if (pid < 0) {
  printf("Fork Failed\n");
exit(1);
}
}
printf("Parent process = PID : %d\n", getpid());
return 0;
}
mahnoor@mahnoor-VMware-Virtual-Platform:~$
```

```
}
}
printf("Parent process = PID : %d\n", getpid());
return 0;
}
mahnoor@mahnoor-VMware-Virtual-Platform:~$ gcc c2.c -o c2
mahnoor@mahnoor-VMware-Virtual-Platform:~$ ./c2
Child 2 = PID : 3734, PPID : 3732
Child 1 = PID : 3733, PPID : 3732
Child 3 = PID : 3735, PPID : 3732
Parent process = PID : 3732
mahnoor@mahnoor-VMware-Virtual-Platform:~$
```

Task 3:

System Calls

A system call is a way for a program to communicate with the Operating System (OS). Whenever a user program wants to do something that needs help from the OS like creating a process, reading a file, or using a printer it does so through a system call.

Types of System Calls:

Type	Purpose	Examples
Process Control	**To create, manage or end processes**	**fork(), exec (), wait(), exit()**
File Management	**To handles files like creating, read, write etc**	**create (), open (), close(), etc.**
Device Management	**To access hardware devices**	**(this is generally mentioned in lab but can't find examples for it)**
Information Maintenance	**To get system or process info**	**getpid(), getppid()**

Few Examples:

Fork () : → Used to create a new child process.

```
pid_t pid = fork();
```

exit(0) : → Ends a process and returns a value to the parent.

getpid() and getppid() : → Used to get the current process ID and the parent process ID

```
#include <stdio.h>
#include <unistd.h>
int main(void) {
   printf("The PID of this process = %d\n", getpid());
   printf("The PID of Parent process = %d\n", getppid())
 return 0; }
```