Kalyani Government Engineering College

Department of Computer Science & Engineering

OS-LAB : CS-693

Group No. :

Name	Roll
Rounak Polley	10200115033
Rounak Mukherjee	10200115032
Rumeet Prasad	10200115034
Ritik Raj	10200115031

Lab assignment: 2

1. Create a process using fork () using C code.

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

int main(){
        int id;
        printf("Creating Process using fork()\n");
        printf("Process id of program process : %d \n", getpid());
        printf("Process id of it's parent : %d \n", getppid());
        id = fork();
        printf("id = %d \n",id);
        if(id>0){
                printf("Parent section PID : %d \n", getpid());
        }
        else if(id==0){
                printf("Process created using fork PID : %d - from Parent PID : %d \n",getpid(),getppid());
        }
        else{
                printf("fork creation failed \n");
        }
        return 0;
}
```

OUTPUT

```
Process id of program process : 2111
Process id of it's parent : 2030
id = 2112
Parent section PID : 2111
id = 0
Process created using fork PID : 2112 - from Parent PID : 2111
```

2. Create an Orphan process using C Code.

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

int main(){
        int pid;
        printf("This Is The Parent Process\n");
        printf("0. PIDs : %d %d\n", getpid(), getppid());
        pid = fork();
        if(pid==0){
```

```
                sleep(1);
                printf("This is the child Process\n");
                printf("1. PIDs : %d, %d\n",getpid(), getppid());
        }
        sleep(2);
        printf("2. PIDs : %d, %d\n",getpid(), getppid());
        if(getppid() == 1){
                 printf("Process is now orphan");
        }
        return 0;
}
```

OUTPUT

```
[root@localhost RounakPolley]# ./a.out orphan.c
This Is The Parent Process
0. PIDs : 1653 1315
This is the child Process
1. PIDs : 1654, 1653
2. PIDs : 1653, 1315
[root@localhost RounakPolley]# 2. PIDs : 1654, 1
Process is now orphan
```

3. Kill the MP3 playing process on your system.

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

int main(){
        printf("Killing this process : %d\n",getpid());
        sleep(2);
        kill(getpid(),SIGKILL);
        printf("This will not be displayed\n");
        return 0;
}
```

OUTPUT

```
[root@localhost RounakPolley]# ./a.out process_kill.c
Killing this process : 1807
Killed
```

4. Write a program that uses the child to compute partial sums and parent to compute the partial products of an array of integers.

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/wait.h>

int main(){
        int a,b,status;
        pid_t id;
        printf("Enter two numbers\n");
        scanf("%d %d",&a,&b);
        id=fork();
        if(id==0)
        {
                int w=wait(&status);
                printf("a + b = %d from child PID : %d \n",(a+b),getpid());
        }
        else
        {
                printf("a * b = %d from parent PID : %d \n",(a*b),getpid());
        }
        return 0;
}
```

OUTPUT

```
Enter two numbers
12
13
a * b = 156 from parent PID : 2143
a + b = 25 from child PID : 2144
```

5. Create a Zombie process using C Code.

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

int main(){
        printf("ZOMBIE PROCESS\n");
        pid_t pid;
        pid = fork();
        /* Child */
        if (pid == 0){
                printf("This is the child process : %d %d\n",getpid(), getppid());
                exit(0);
                //kill(getpid(),SIGKILL);
        }
        /* Parent... */
        sleep(2);
        printf("This is the parent who's child is a zombie : %d\n",getpid());
        return 0;
}
```

OUTPUT

```
ZOMBIE PROCESS
This is the child process : 2016 2015
This is the parent who's child is a zombie : 2015
```

6. Demonstrate multiple fork() operation using C code

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

int main(){
        int id;
        printf("Creating Process using fork()\n");
        printf("Process id of program process : %d \n", getpid());
        printf("Process id of it's parent : %d \n", getppid());
        id = fork();
        printf("id = %d \n",id);
        if(id>0){
                printf("Parent section PID : %d \n", getpid());
        }
        else if(id==0){
                printf("Process created using fork PID : %d - from Parent PID : %d \n",getpid(),getppid());
        }
        else{
                printf("fork creation failed \n");
        }
        return 0;
}
```

OUTPUT

```
Process id of program process : 2111
Process id of it's parent : 2030
id = 2112
Parent section PID : 2111
id = 0
Process created using fork PID : 2112 - from Parent PID : 2111
```

7. Write a code to show child process executes before parent process using wait ().

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/wait.h>
#include<signal.h>

int main(){
        printf("Child extecutes before parent using wait()\n");
        pid_t pid;
        int status;
        pid = fork();
        if(pid == 0){
                printf("Child : %d , it's Parent: %d\n",getpid(),getppid());
        }
        else{
                int w = wait(&status);
                printf("Parent: %d\n",getpid());
        }
        return 0;
}
```

OUTPUT

```
Child extecutes before parent using wait()
Child : 2104 , it's Parent: 2103
Parent: 2103
```