Practical No.1

Name: Jadhav Siddhesh Ramesh
Class: T.Y.BSc(Comp. Science)
Roll No: 32
Div: A Batch: B
Subject : Operating System-I
Assignment Name: Operating System Practical CS-357
Performance Date: Submission Date:

Set-A

Q.1) Implement the C Program to create a child process using fork(), display parent and child process id. Child process will display the message "I am Child Process" and the parent process should display "I am Parent Process".

Program:

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
int pid=fork();
if(pid>0)
{
printf("I am parent process\n");
printf("ID:%d\n\n",getpid());
}
else if(pid==0)
{
printf("I am child process\n");
printf("ID:%d\n",getpid());
}
else
{
printf("Failed to create child process");
}
return 0;
}
```

Output:

```
I am parent process
ID:2392

I am child process
ID:2393
```

Q.2) Write a program that demonstrates the use of nice() system call. After a child process is started using fork(), assign higher priority to the child using nice() system call.

Program:

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

```
#include<unistd.h>
int main()
{
pid_t pid;
pid=fork();
if(pid==0)
{
printf("\nI am child process, id=%d\n",getpid());
printf("\nPriority:%d,id=%d\n",nice(-7),getpid());
}
else
{
printf("\nI am parent process, id=%d\n",getpid());
nice(1);
printf("\nPriority:%d,id=%d\n",nice(15),getpid());
}
return 0;
}
```

Output:

```
I am parent process, id=2521

Priority:16,id=2521

I am child process, id=2522

Priority:-1,id=2522
```

Set-B

Q.1) Write a C program to illustrate the concept of orphan process. Parent process creates a child and terminates before child has finished its task. So child process becomes orphan process. (Use fork(), sleep(), getpid(), getppid()).

Program:

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
int pid;
pid=getpid();
printf("Current Process ID is:%d\n",pid);
printf("\n[Forking Process...]\n");
pid=fork();
if(pid<0)
{
printf("\n Process can not be created");
}
else
{
```

```
if(pid==0)
{
printf("\nChild Process is Sleeping..");
sleep(5);
printf("\n Orphan Child's Parent ID:%d",getpid());
}
else
{
/*Parent Process*/
printf("\n Parent Process  Completed...");
}
}
return 0;
}
```

Output:

```
Current Process ID is:2331

[Forking Process...]

 Parent Process  Completed...
 Child Process is Sleeping..
 Orphan Child's Parent ID:2332
```