21BAI1217
MAINAK CHATTOPADHYAY

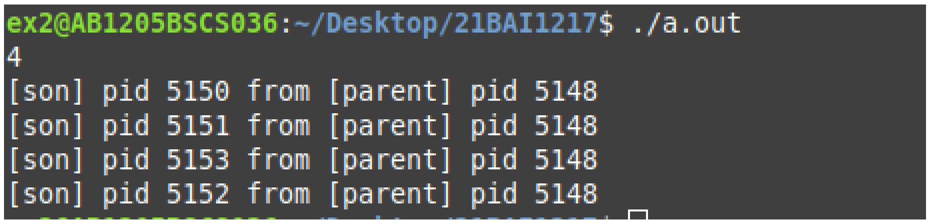
OPERATING SYSTEMS LAB

1.Write a C program to generate a child process using 4 system call. You can generate any number of child process as required. Get the number of child process to be generated as a input from the user and those child process should be generated for the same parent process. Display the child process and the parent process ID(use getpid() for child process ID and getppid() for parent process ID).

CODE

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

int main()
{
        int n;
        scanf("%d",&n);
        for(int i=0;i<n;i++) // loop will run n times (n=4)
        {
                if(fork() == 0)
                {
                        printf("[son] pid %d from [parent] pid %d\n",getpid(),getppid());
                        exit(0);
                }
        }
        for(int i=0;i<5;i++) // loop will run n times (n=4)
        wait(NULL);

}
```

OUTPUT



```
ex2@AB1205BSCS036:~/Desktop/21BAI1217$ ./a.out
4
[son] pid 5150 from [parent] pid 5148
[son] pid 5151 from [parent] pid 5148
[son] pid 5153 from [parent] pid 5148
[son] pid 5152 from [parent] pid 5148
```

2. Write a C program to illustrate process termination.

a)Create five process(1 parent process and 4 child process) where they should terminate atlast after all the child processes are terminated.

b) fourth child process terminates last before the parent process.

c) first child process terminates after the third child process.

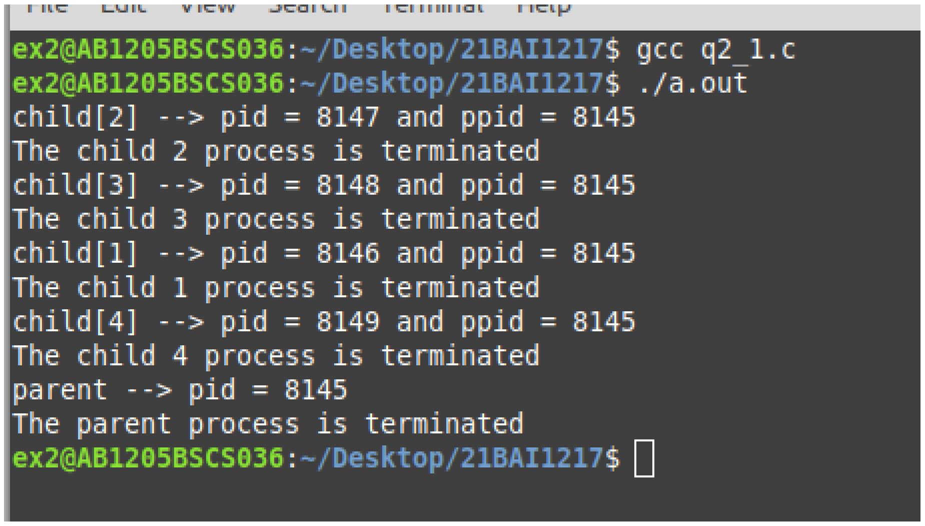
CODE -

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

#include<sys/wait.h>


int main()

{

        int pid, pid1, pid2,pid3;

        pid = fork();

        if (pid == 0) {

                sleep(3);

                printf("child[1] --> pid = %d and ppid = %d\n",

                        getpid(), getppid());

                printf("The child 1 process is terminated\n");

        }
```

```
else {
        pid1 = fork();
        if (pid1 == 0){
                sleep(1);
                printf("child[2] --> pid = %d and ppid = %d\n",
                        getpid(), getppid());
                        printf("The child 2 process is terminated\n");
        }
        else{
                pid2 = fork();
                if (pid2 == 0) {
                        sleep(2);
                        printf("child[3] --> pid = %d and ppid = %d\n",
                                getpid(), getppid());
                        printf("The child 3 process is terminated\n");
                }
                else{
                        pid3 = fork();
                        if(pid3 == 0){
                                sleep(4);
                                printf("child[4] --> pid = %d and ppid = %d\n",
                                        getpid(), getppid());
                                printf("The child 4 process is terminated\n");
                        }
```

```
                    else {
                              sleep(5);
                              printf("parent --> pid = %d\n", getpid());
                              printf("The parent process is terminated\n");
                    }
          }
     }
   }
   for(int i=0;i<4;i++){
          wait(NULL);
   }
   exit(0);
   return 0;
}
```

OUTPUT -



```
ex2@AB1205BSCS036:~/Desktop/21BAI1217$ gcc q2_1.c
ex2@AB1205BSCS036:~/Desktop/21BAI1217$ ./a.out
child[2] --> pid = 8147 and ppid = 8145
The child 2 process is terminated
child[3] --> pid = 8148 and ppid = 8145
The child 3 process is terminated
child[1] --> pid = 8146 and ppid = 8145
The child 1 process is terminated
child[4] --> pid = 8149 and ppid = 8145
The child 4 process is terminated
parent --> pid = 8145
The parent process is terminated
ex2@AB1205BSCS036:~/Desktop/21BAI1217$
```