Name: Meet Brijwani
Roll no: 14
Batch: S11

Experiment 4: Linux - Process

Aim:

a. To create a child process in Linux using the fork system call.From the child process obtain the process ID of both child and parent by using getpid and getpid system call.

b. Explore wait and waitpid before termination of process.

Theory:

1] System call

When a program is in user mode and requires access to Ram or hardware resources, it must ask the kernel to provide access to that resource. This is done via something called a system call. When a program makes a system call, the mode is switched from user mode to Kernel mode. This is called context switch. The kernel provides the resources which the program requested. System calls are made by user level programmes in following cases:

Creating, opening, closing and delete files in the system
Creating and managing new processes
Creating a connection in the network, sending and receiving packets
Requesting access to a hardware device like a mouse or a printer

2] Fork ( )

The fork system call is used to create processes. When a process makes a fork().call, an exact copy of the process is created.

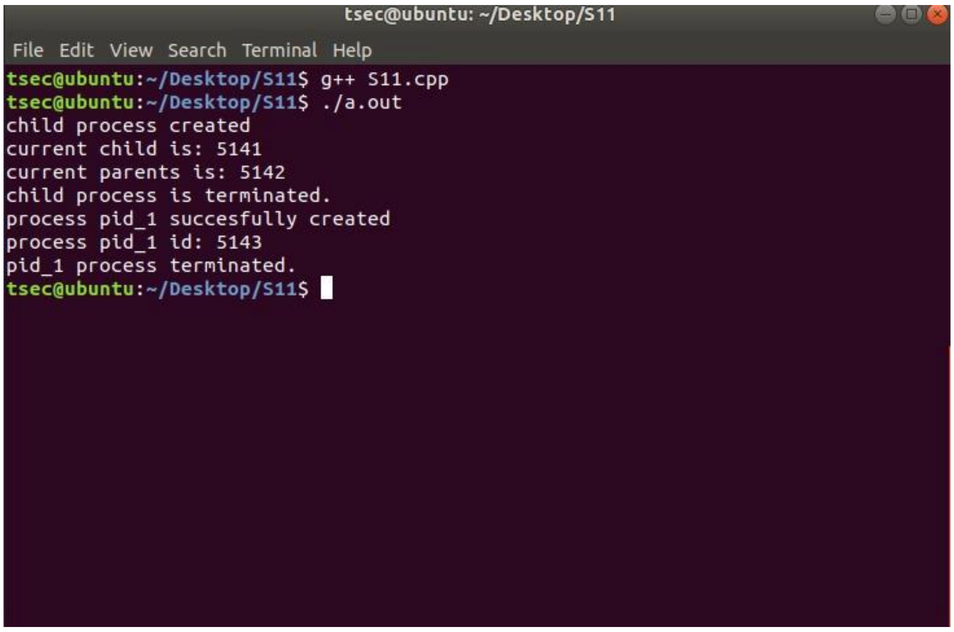
There are now two processes, one being the parent process and the other being the child process. The process which called fork()call is the parent process and the process which is created newly is called child process. The child process will be exactly the same as the parent. The process state of the parent i.e the address space, variables, open files etc is copied into the child process. The change of values in the parent process doesn’t affect the child and vice versa.

Code:

```
#include <iostream>
#include<sys/wait.h>
#include<unistd.h>

using namespace std;
int wait_func()
{
  int pid_1 = fork();
if(pid_1==0)
  {     cout<<"process pid_1 succesfully
created"<<"\n";     cout<<"process pid_1 id: "<<
getpid()<<"\n";     exit(0);   }
  waitpid(pid_1, NULL,0);
  cout << "pid_1 process terminated."<<"\n";
return 0;
}
int main()
{
  int pid=fork();
if(pid==0)
  {
   cout<<"child process created"<<"\n";
cout<<"current child is: "<<getppid()<<"\n";
cout<<"current parents is: "<<getpid()<<"\n";
exit(0);   }   wait(NULL);   cout<<"child process
is terminated."<<"\n";
  wait_func();
  return 0;
}
```

Output:



```
tsec@ubuntu:~/Desktop/S11$ g++ S11.cpp
tsec@ubuntu:~/Desktop/S11$ ./a.out
child process created
current child is: 5141
current parents is: 5142
child process is terminated.
process pid_1 succesfully created
process pid_1 id: 5143
pid_1 process terminated.
tsec@ubuntu:~/Desktop/S11$ 
```

Conclusion:

Thus, we have successfully studied and implemented how to create child process in Linux using fork() system calls.