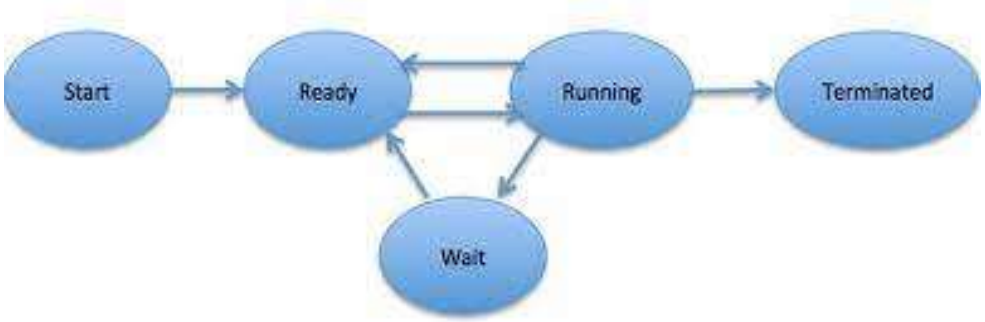
Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department &
Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** OS

Name	**Pratik Pujari**		
UID no.	**2020300054**	**Class:**	**Comps C Batch**
Experiment No.	3		

AIM:	To evaluate the concepts of fork method
THEORY:	**What is a process?** A process is the instance of a computer program that is being executed by one or many threads. It contains the program code and its activity. Depending on the operating system (OS), a process may be made up of multiple threads of execution that execute instructions concurrently. • Whenever a command is issued in Unix/Linux, it creates/starts a new process. For example, pwd when issued which is used to list the current directory location the user is in, a process starts. • Through a 5 digit ID number Unix/Linux keeps an account of the processes, this number is call process ID or PID. Each process in the system has a unique PID. • Used up pid's can be used in again for a newer process since all the possible combinations are used. (tutorialpoints.com) 

Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department &
Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** OS

Initializing a process

A process can be run in two ways:
Method 1: Foreground Process: Every process when started runs in foreground by default, receives input from the keyboard, and sends output to the screen. When issuing pwd command
$ ls pwd
Output:
$ /home/username/root
When a command/process is running in the foreground and is taking a lot of time, no other processes can be run or started because the prompt would not be available until the program finishes processing and comes out.

Method 2: Background Process: It runs in the background without keyboard input and waits till keyboard input is required. Thus, other processes can be done in parallel with the process running in the background since they do not have to wait for the previous process to be completed. Adding & along with the command starts it as a background process
$ pwd &
Since pwd does not want any input from the keyboard, it goes to the stop state until moved to the foreground and given any data input. Thus, on pressing Enter:
Output:
[1] + Done pwd
$
That first line contains information about the background process – the job number and the process ID. It tells you that the ls command background process finishes successfully. The second is a prompt for another command.

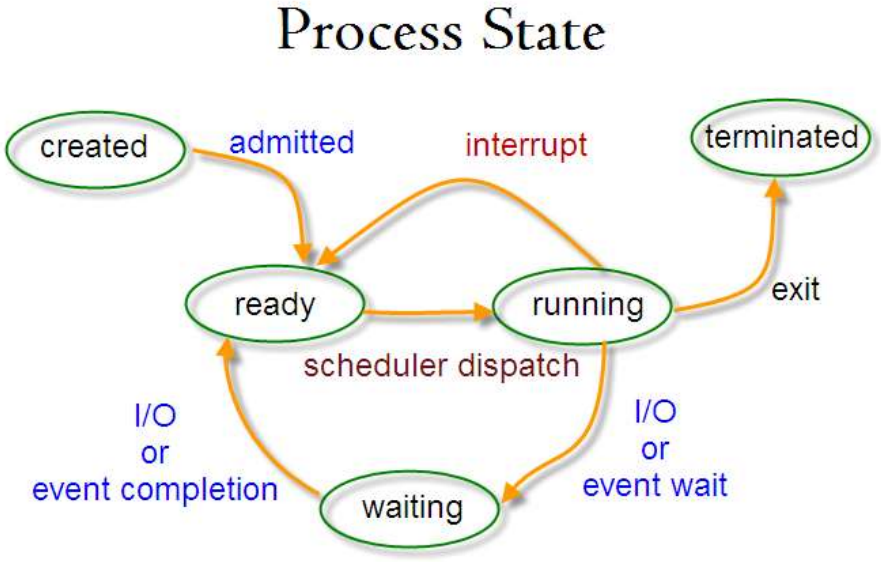
Types of Processes

Computer Engineering Department &
Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** OS

1. **Parent and Child process :** The 2nd and 3rd column of the ps –f command shows process id and parent's process id number. For each user process, there's a parent process in the system, with most of the commands having shell as their parent.
2. **Zombie and Orphan process :** After completing its execution a child process is terminated or killed and SIGCHLD updates the parent process about the termination and thus can continue the task assigned to it.
3. **Daemon process :** They are system-related background processes that often run with the permissions of root and services requests from other processes.

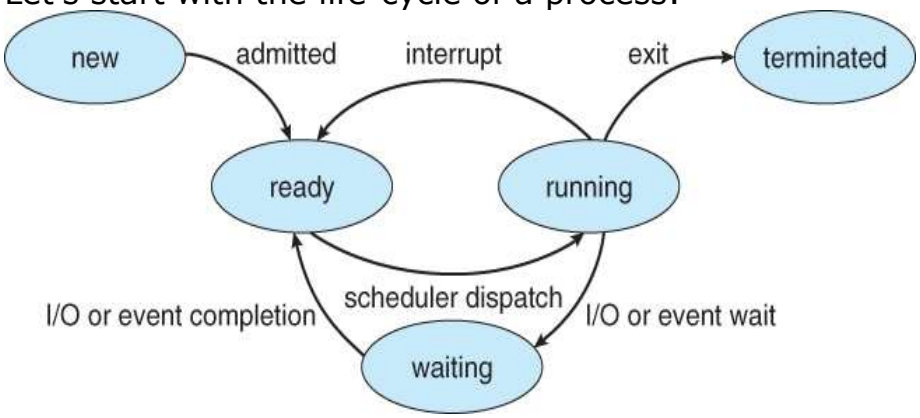


What is a Fork()?

In the computing field, **fork()** is the primary method of process creation on Unix-like operating systems. This function creates a new copy called the child out of the original process, that is called the parent. When the parent

process closes or crashes for some reason, it also kills the child process.

Let's start with the life-cycle of a process:



The operating system is using a unique id for every process to keep track of all processes. And for that, fork() doesn't take any parameter and return an int value as following:

- Zero: if it is the child process (the process created).
- Positive value: if it is the parent process.
- Negative value: if an error occurred.

PID

A process ID (PID) is a unique identifier assigned to a process while it runs. When the process ends, its PID is returned to the system. Each time you run a process, it has a different PID (it takes a long time for a PID to be reused by the system). You can use the PID to track the status of a process with the **ps** command or the **jobs** command, or to end a process with the **kill** command.

PPID

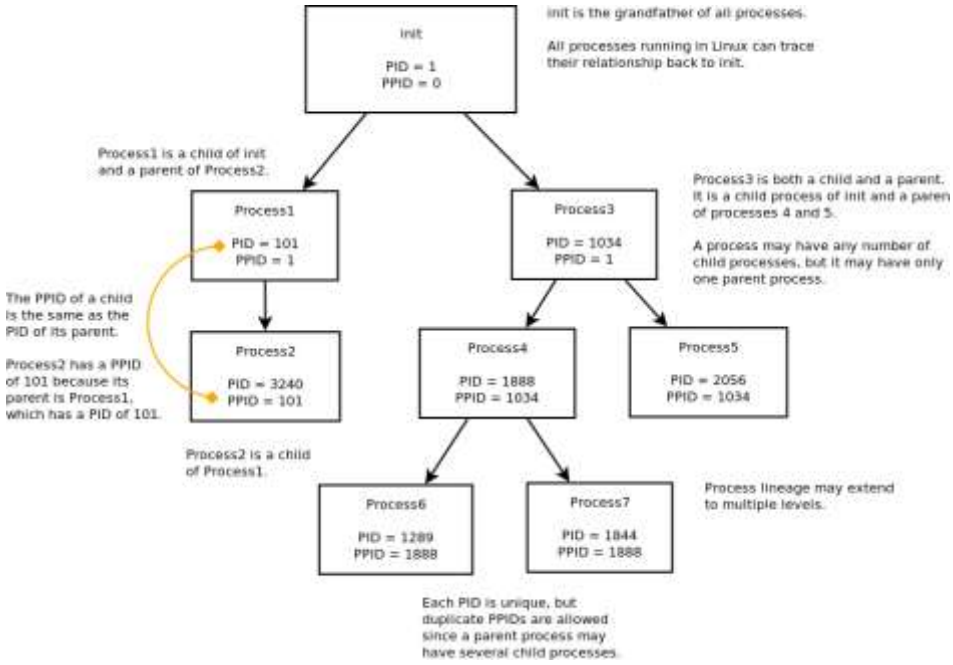
A process that creates a new process is called a *parent process*; the new process is called a *child process*. The parent process ID (PPID) becomes associated with the new child process when it is created. The PPID is not used for job control.

Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department & Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** OS



EXPERIMENT 1

CODE:

```
#include<stdio.h>
#include<unistd.h>
int main(){
        int p_id,pp_id;
        static int counter=0;
        for(int i=0;i<=1;i++){
                int id1=fork();
                int id2=fork();
                p_id=getpid();
                pp_id=getppid();
                printf("\nFork 1:%d\tProcess Id: %d\tParent
Process Id: %d",id1,p_id,pp_id);
                printf("\nFork 2:%d\tProcess Id: %d\tParent
Process Id: %d\n",id2,p_id,pp_id);

        }
        return 0;}
```

Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department & Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** OS

OUTPUT:

```
os-lab@sem4:~/Desktop/Programs/EXP 3$ ./a.out

Fork 1:4024     Process Id: 4022        Parent Process Id: 3883
Fork 2:4025     Process Id: 4022        Parent Process Id: 3883

Fork 1:4026     Process Id: 4022        Parent Process Id: 3883
Fork 2:4027     Process Id: 4022        Parent Process Id: 3883
os-lab@sem4:~/Desktop/Programs/EXP 3$
Fork 1:4024     Process Id: 4025        Parent Process Id: 3233
Fork 2:0        Process Id: 4025        Parent Process Id: 3233



Fork 1:4028     Process Id: 4025        Parent Process Id: 3233
Fork 2:4029     Process Id: 4025        Parent Process Id: 3233

Fork 1:4026     Process Id: 4027        Parent Process Id: 3233
Fork 2:0        Process Id: 4027        Parent Process Id: 3233
Fork 1:0        Process Id: 4026        Parent Process Id: 3233
Fork 2:4030     Process Id: 4026        Parent Process Id: 3233
Fork 1:0        Process Id: 4024        Parent Process Id: 3233
Fork 2:4031     Process Id: 4024        Parent Process Id: 3233




Fork 1:0        Process Id: 4030        Parent Process Id: 3233
Fork 2:0        Process Id: 4030        Parent Process Id: 3233
Fork 1:4032     Process Id: 4024        Parent Process Id: 3233
Fork 2:4033     Process Id: 4024        Parent Process Id: 3233
Fork 1:0        Process Id: 4028        Parent Process Id: 3233
Fork 2:4034     Process Id: 4028        Parent Process Id: 3233
Fork 1:0        Process Id: 4031        Parent Process Id: 4024
Fork 2:0        Process Id: 4031        Parent Process Id: 4024

Fork 1:4028     Process Id: 4029        Parent Process Id: 3233
Fork 2:0        Process Id: 4029        Parent Process Id: 3233
Fork 1:4035     Process Id: 4031        Parent Process Id: 3233
Fork 2:4036     Process Id: 4031        Parent Process Id: 3233


Fork 1:4032     Process Id: 4033        Parent Process Id: 3233
Fork 2:0        Process Id: 4033        Parent Process Id: 3233
Fork 1:0        Process Id: 4034        Parent Process Id: 3233
Fork 2:0        Process Id: 4034        Parent Process Id: 3233
Fork 1:0        Process Id: 4032        Parent Process Id: 3233
Fork 2:4037     Process Id: 4032        Parent Process Id: 3233
```

Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department & Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** OS

```
Fork 1:4035     Process Id: 4036        Parent Process Id: 3233
Fork 2:0        Process Id: 4036        Parent Process Id: 3233
Fork 1:0        Process Id: 4035        Parent Process Id: 3233
Fork 2:4038     Process Id: 4035        Parent Process Id: 3233

Fork 1:0        Process Id: 4037        Parent Process Id: 3233
Fork 2:0        Process Id: 4037        Parent Process Id: 3233

Fork 1:0        Process Id: 4038        Parent Process Id: 3233
Fork 2:0        Process Id: 4038        Parent Process Id: 3233
```

Explanation: Two forks are written are for loop and it runs twice which makes the four forks. So 2^4 part out of which child and parent are assigned pid and ppid.

Example of fork():

```
fork ();   // Line 1
fork ();   // Line 2
fork ();   // Line 3

       L1       // There will be 1 child process
    /     \     // created by line 1.
  L2      L2    // There will be 2 child processes
 /  \    /  \   //  created by line 2
L3  L3  L3  L3  // There will be 4 child processes
                // created by line 3
```

CONCLUSION: Learnt about the fork method and using the unistd.h and importing the functions getpid(); and getppid(); to get the process and parent process id. Learnt that different threads are assigned to the forks of child and parents and are executed accordingly