JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD

Lecture-21 By Vivek Dubey

COMPUTER SCIENCE AND ENGINEERING

CS403PC

COMPUTER SCIENCE AND ENGINEERING (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

CS403PC

BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY

Scheme of Operating System

S. No.	Course Code	Course Title	L	T	P	Credits
3	CS403PC	Operating Systems	3	0	0	3
6	CS406PC	Operating Systems Lab	0	0	3	1.5

Lecture-21 By Vivek Dubey

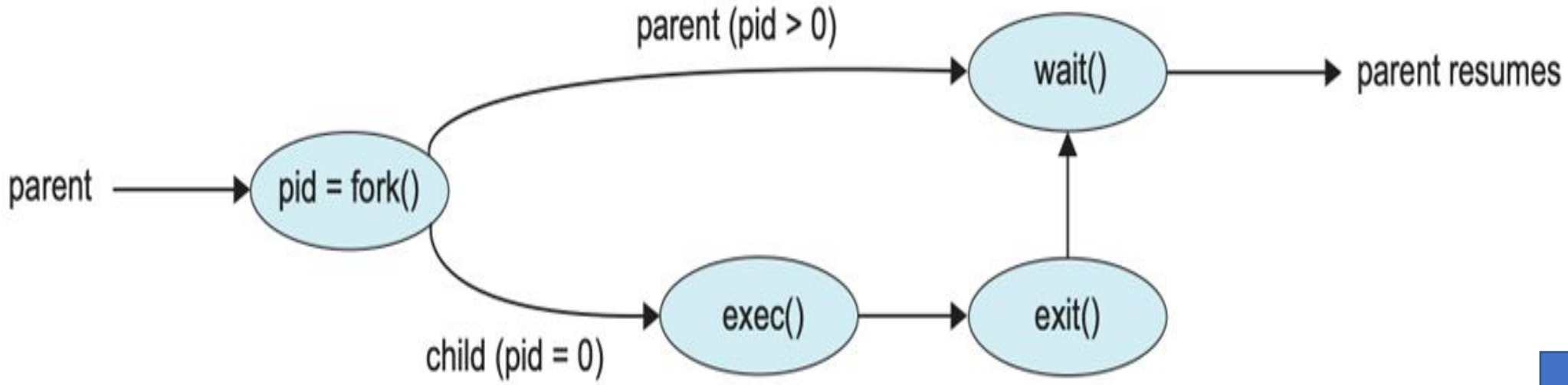
UNIT-2:

- Process and CPU Scheduling –
  - Process concepts and scheduling, Operations on Processes, Cooperating Processes, Threads, and Interposes Communication, Scheduling Criteria, Scheduling Algorithms, Multiple-Processor Scheduling.
- System call interface for process management-
  - fork, exit, wait, waitpid, exec

Lecture-21 By Vivek Dubey

System call interface for process management

- fork,
- exit,
- wait,
- waitpid,
- exec



Contd..

- fork
  - A parent process uses fork to create a new child process. The child process is a copy of the parent. After fork, both parent and child executes the same program but in separate processes.
- exec
  - Replaces the program executed by a process. The child may use exec after a fork to replace the process' memory space with a new program executable making the child execute a different program than the parent.
- exit
  - Terminates the process with an exit status.
- wait
  - The parent may use wait to suspend execution until a child terminates. Using wait the parent can obtain the exit status of a terminated child.

Find Process id


```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
    printf("Hello Vivek Dubey!\n PID=%d\n",getpid());
    return 0;
}
```

Run-1
Hello Vivek Dubey!
PID=2389

Run-2
Hello Vivek Dubey!
PID=1718

Run-3
Hello Vivek Dubey!
PID=1274

Create Process using fork()


```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
    fork();
    printf("Hello Vivek Dubey!\n PID=%d\n",getpid());
    return 0;
}
```

Run-1
Hello Vivek Dubey!
PID=188
Hello Vivek Dubey!
PID=192

Run-2
Hello Vivek Dubey!
PID=2443

Verify Child Process


```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
    int fpid=fork();
    printf("Hello Vivek Dubey!\n PID=%d fpid=%d\n",getpid(),fpid);
    return 0;
}
```

Run-1
Hello Vivek Dubey!
PID=1743 fpid=1747
Hello Vivek Dubey!
PID=1747 **fpid=0**

Run-2
Hello Vivek Dubey!
PID=1940 **fpid=1944**

Conclusion

- Case - 1
- if(fpid < 0) then child was not created successfully and return 1;

- Case - 2
- if(pid == 0) then child process will be created

- Case – 3
- If(fpid > 0) then Parent process code will run only

How many child process will be created?

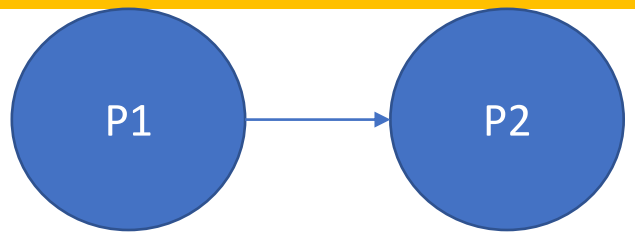
7

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
    fork();
    fork();
    printf("Hello Vivek Dubey!\n PID=%d\n",getpid());
    return 0;
}
```

```
Hello Vivek Dubey! (1)
 PID = 107
Hello Vivek Dubey! (2)
 PID = 106
Hello Vivek Dubey! (3)
 PID = 108
Hello Vivek Dubey! (4)
 PID = 110
Hello Vivek Dubey! (5)
 PID = 111
Hello Vivek Dubey! (6)
 PID = 105
Hello Vivek Dubey! (7)
 PID = 112
localhost:~/Vivek# Hello Vivek Dube
y! (8)
 PID = 109
```

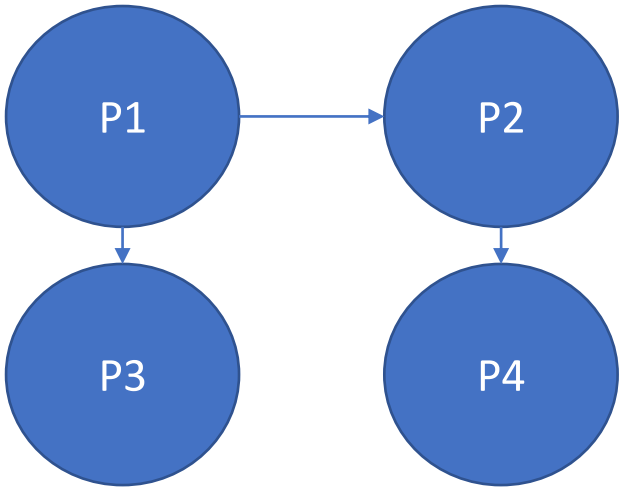
How 8 process will be created?

Step-1

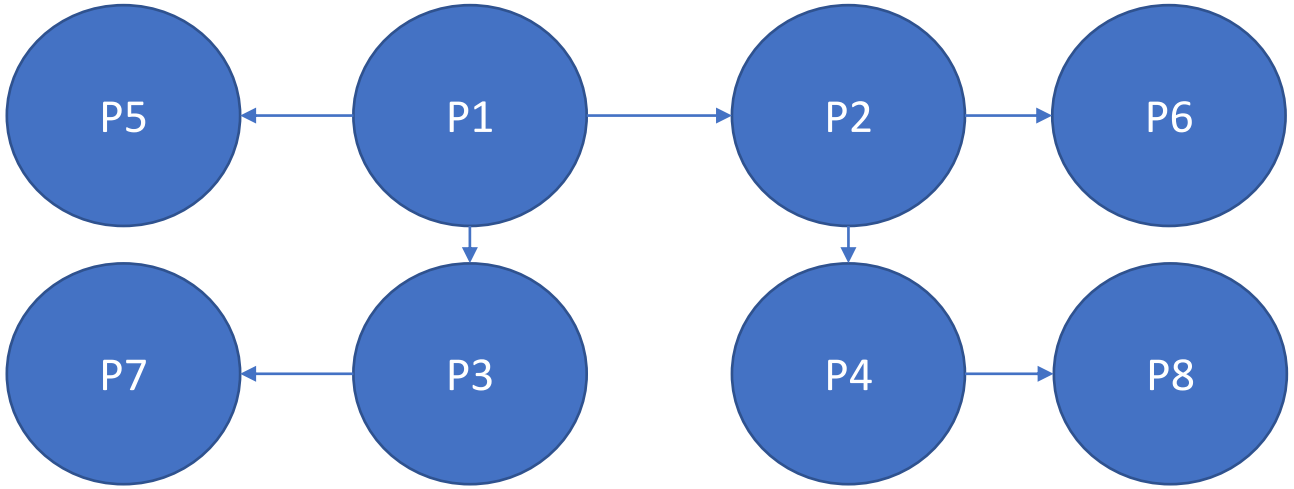


Step-2
Using 1st fork()

Step-3
Using 2nd fork()



Step-4
Using 4th fork()



8

Lecture-21 By Vivek Dubey

Use exec() system call (Part-1)


```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
        printf("PID of ex1.c = %d\n", getpid());
        char *args[] = {"Hello", "Vivek", "Dubey", NULL};
        execv("./ex2", args);
        printf("Back to ex1.c");
}
```

Part-2


```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main(int args, char *argv[])
{
        printf("We are in ex2.c\n");
        printf("PID of ex2.c = %d\n", getpid());
        return 0;
}
```

Run

```
localhost:~/Vivek# cc ex1.c -o ex1
localhost:~/Vivek# cc ex2.c -o ex2
localhost:~/Vivek# ./ex1
PID of ex1.c = 161
We are in ex2.c
PID of ex2.c = 161
```

Using wait()


```
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<time.h>
#include<stdio.h>
int main(int argc, char* argv[])
{
        int id = fork();
        int n;
        if (id == 0)
        { n = 1;}
        else
        { n = 6;}

        int i;
        for( i = n; i < n + 5; i++)
        {
                printf("%d",i);
        }
        printf("\n");
        return 0;
}
```

```
Run
localhost:~/Vivek# cc ex3.c
localhost:~/Vivek# ./a.out
678910
12345
```

Part-2


```
Run
localhost:~/Vivek# cc ex3.c
localhost:~/Vivek# ./a.out
12345
678910
```

```
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<time.h>
#include<stdio.h>
int main(int argc, char* argv[])
{
        int id = fork();
        int n;
        if (id == 0){ n = 1;}
        else { n = 6;}

        if (id !=0) { wait(); }

        int i;
        for( i = n; i < n + 5; i++)
        {
                printf("%d",i);
        }
        printf("\n");
        return 0;
}
```