

PRACTICAL 2

Write a program using the following system calls of UNIX operating system: fork, exec, getpid, exit, wait, stat, readdir, opendir.

Aim:

Task-01 Write a program to execute fork() and find out the process id by getpid() system call.

Program:

File 1: prac2.c

```
GNU nano 5.3          prac2.c
//execute fork() and exec() system calls and find process id with the getpid() system call

#include<stdio.h>
#include<unistd.h>

void main(){

    fork();
    printf("Hello World\n");
    printf("Child Process ID: %d\nParent Process ID: %d\n",getpid(),getppid());
    printf("I am xyz.c called by execvp()\n");
    printf("I am xyz.c called by execv()\n");
    printf("I am xyz.c called by execlp()\n");
    printf("I am xyz.c called by execvpe()\n");
    printf("I am xyz.c called by execl()\n");
    printf("I am xyz.c called by execl()\n");
}
```

File 2: xyz.c

```
GNU nano 5.3          xyz.c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>

int main(){

    char *args[]={ "./prac2",NULL};
    execvp(args[0],args);
    execv(args[0],args);
    execlp(args[0],args);
    execvpe(args[0],args);
    execl(args[0],args);
    execl(args[0],args);
    printf("End-----\n");
    return 0;
}
```

Output:

```
[root@localhost ~]# ls
bench.py  hello.c  prac2  prac2.c  xyz  xyz.c
[root@localhost ~]# ./xyz
Hello World
Child Processs ID: 210
Parent Process ID: 48
I am xyz.c called by execvp()
I am xyz.c called by execv()
I am xyz.c called by execlp()
I am xyz.c called by execvpe()
I am xyz.c called by execl()
I am xyz.c called by execl()
Hello World
Child Processs ID: 211
Parent Process ID: 210
I am xyz.c called by execvp()
I am xyz.c called by execv()
I am xyz.c called by execlp()
I am xyz.c called by execvpe()
I am xyz.c called by execl()
I am xyz.c called by execl()
[root@localhost ~]#
```

Conclusion:

In this practical we have learnt about fork(), getpid() and getppid(), exec() system calls.