

Simple Program

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
    fork();
    printf("Called fork() system call\n");
    return 0;
}
```

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
    pid_t pid,mypid,myppid;
    pid=getpid();
    printf("Before fork(): Process id is%\n",pid);
    pid=fork();
    if(pid<0)
        perror("fork() failure\n");
        return 1;
    }
    // child process
    if(pid==0)
    {
        printf("This is child Process\n");
        mypid=getpid();
        myppid=getppid();
        printf("Process id is %d and PPID is %d\n",mypid,myppid);
    }else { //parent process
        sleep(2);
        printf("This is parent process\n");
        mypid=getpid();
        myppid=getppid();
        printf("process id is %d and PPID is %d\n",mypid,myppid);
        printf("Newly created process id or child pid is %d\n",pid);
    }
    return 0;
}
```

EXECLP System call :

```
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
int main()
{
    int pid;
    /* fork a child process*/
    pid=fork();
    if(pid<0)
    { /* error occurred */
        fprintf(stderr,"fork Failed");
        return 1;
    }
    else if (pid==0)
    { /* child process */
        execlp("/bin/wc","wc",NULL);
    }
    else { /* parent process */
        /* parent will wait for the child to complete */
        wait(NULL);
        printf("child Complete");
    }
    return 0;
}
```

SIGCLD System Call :

```
#include<stdio.h>
#include<signal.h>
void abc()
int pid
main()
{
    pid=fork();
    if(pid==0)
    else
    {
        signal(SIGCLD,abc); // signal function to handle SIGCLD.signal
        for(i=0;i<1000;i++)
        printf("%d,i)
        print("parent existing \n");
    }
}
void abc()
{
    print("child died \n");
    getchar();
}
```

downloaded from
StudentSuvidha.com

NICE System Call:

```
#include<stdio.h>
main()
{
    int pid,retnice;
    printf("press DEL to stop process \n");
    pid=fork();
    for(;;)
    {
        if(pid==0)
        {
            retnice=nice(-5);
            print(child gets higher CPU priority %d \n",retnice);
            sleep(1);
        }
        else
        {
            retnice=nice(4);
            print("parent gets lower CPU priority %d\n",retnice);
            sleep(1);
        }
    }
}
```

downloaded from
StudentSuvidha.com

Exit() System Call

```
#include<stdio.h>
#include<stdlib.h>
void exitfunc()
{
    printf("Called cleanup function-exitfunc()\n");
    return;
}
int main() {
    atexit(exitfunc);
    printf("Hello,World!\n");
    exit(0);
}
```

EXEC() System Call

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main(int argc,char *argv[])
{
    printf("PID of example.c=%d\n",getpid());
    char *args[]={"Hello","C", "Programming",NULL};
    execv("./hello",args);
    printf("Back to example.c");
    return 0;
}
```