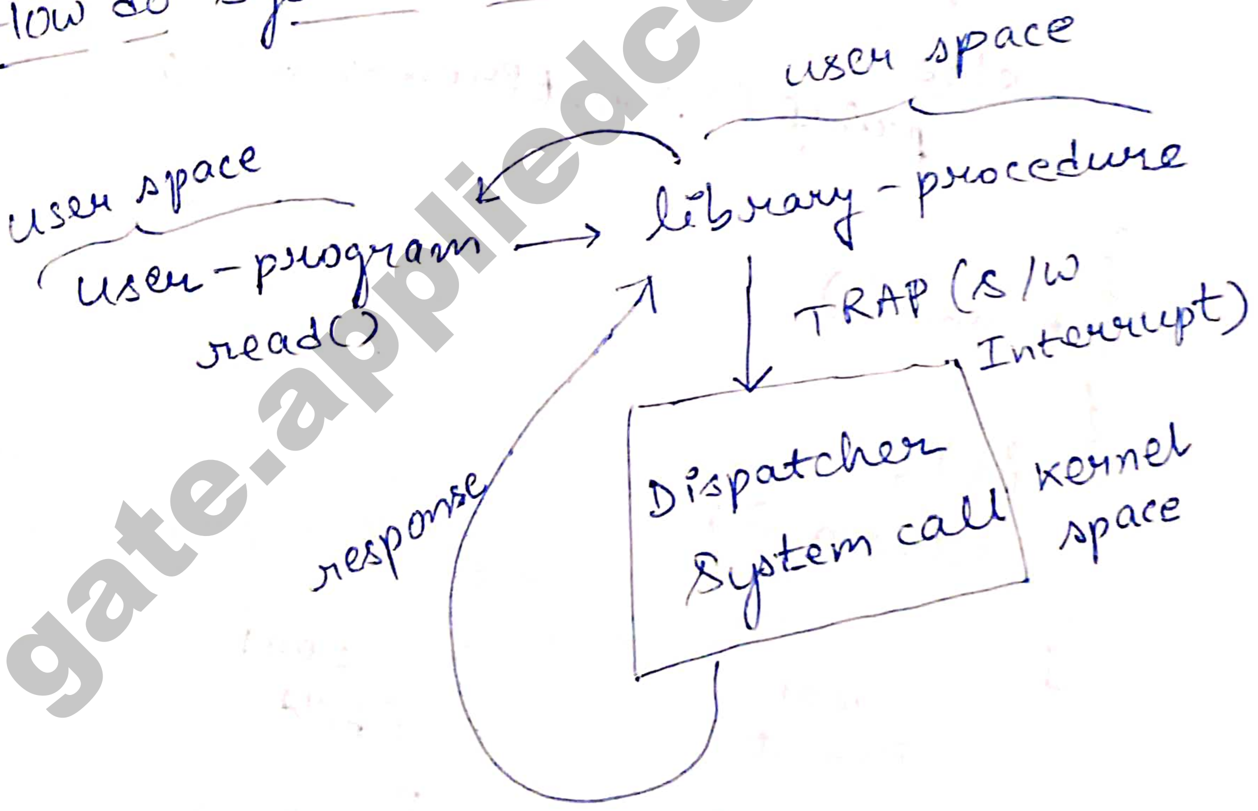
System calls

→ fork() → modes of CPU : kernel & user mode

→ shmget(), shmat() : shared memory in Prod-consumer problems

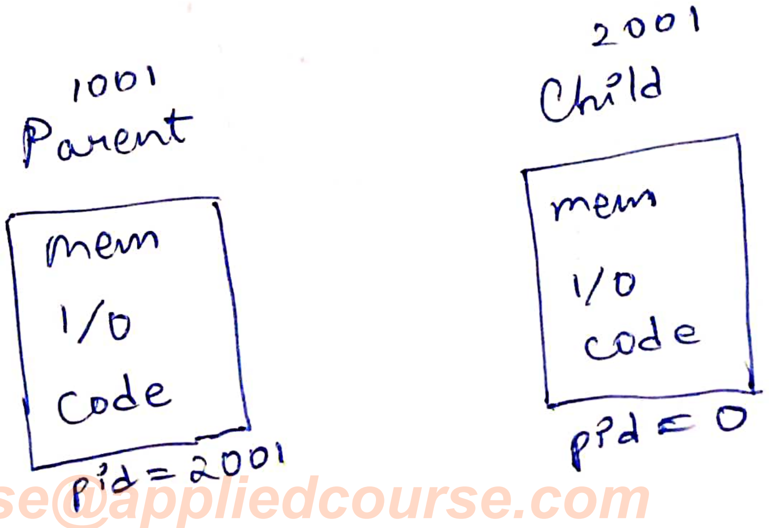
→ more details & examples.

How do system calls work?



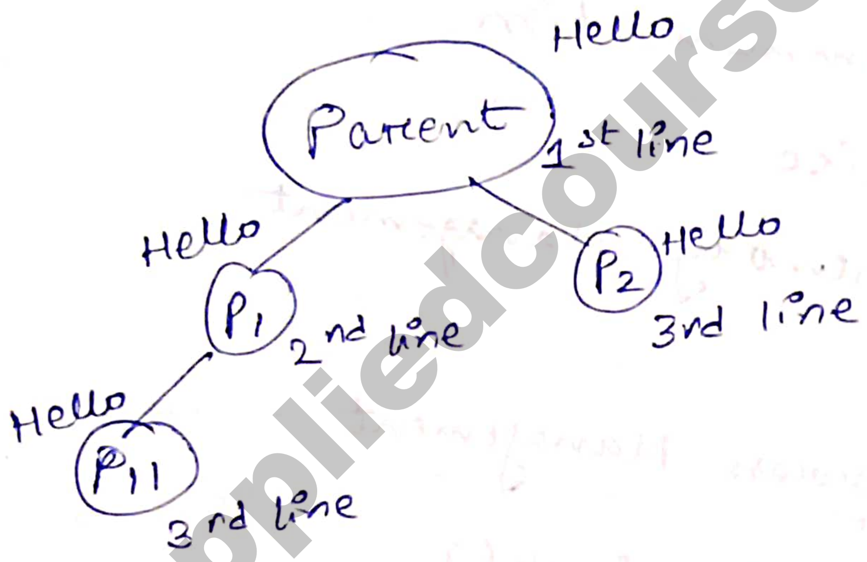
fork() [process management]

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

void f()
{
        int pid = fork();
        if (pid == 0)
           printf("ChildProcess\n");

        else
            printf("ParentProcess\n");

}
int main()
{
    f();
    return 0;
}
```



(Q) # print statements

```
int main()
{
    fork();
    fork();
    printf("Hello\n");
    return 0;
}
```



Note:
Linux has 393 system calls.
We can find them on google
and they are well-documented.

Broadly speaking, system calls can be divided into :

(i) Process Management

(ii) File Management

(iii) Directory & File System

(iv) I/O device management

(v) Communication

(vi) Misc

(vii) Memory Management

I. Process Management

1. pid = fork()

2. pid = waitpid (pid, & statloc, options)

   // wait for a child to terminate

3. s = execve (name, argv, environp)

   // replace a process' core image

4. exit (status)

   // Terminate process execution and return status.

II. File Management

1. fd = open(file, how, ...)
   // Open a file for reading, writing or both.

2. s = close(fd)
   // close an open file

3. n= read(fd, buffer, nbytes)
   // Read data from a file into a buffer.

4. n= write(fd, buffer, nbytes)
   // write data from a buffer into a file.

5. position = lseek(fd, offset, whence)
   // move the file pointer.

6. s=stat(name, & buf)
   // Get a file's status information

III. Directory and File system management

1. s = mkdir (name, mode)
   // create a new directory
2. s=rmdir (name)
   // Remove an empty directory
3. s=link (name1, name2)
   // Create a new entry, name2, pointing to name1.
4. s=unlink (name)
   // Remove a directory entry
5. s = mount(special, name, flag)
   // Mount a file system.
6. s = umount (special)
   // Unmount a file system.

IV. Miscellaneous

1. s = chdir (dirname)
   // Change the working directory

2. s = chmod (name, mode)
// change a file's protection bits

3. s = kill (pid, signal)
// Send a signal to a process.

4. seconds = time (& seconds)
// Get the elapsed time since Jan 1, 1970.

(Q.) system calls are usually invoked by using :
✓(A) A software interrupt
(B) polling
(C) An indirect jump
(D) A priviledged instruction.

Ans :(A) system calls are invoked using a Trap. And Trap is nothing but a software interrupt.

(Q.) (MSQ)

Consider the following statements S1 and S2:

S1: It is important that a programmer knows which library procedures result in system calls.

S2: When an interrupt or a system call transfers control to the operating system, a kernel stack area separate from the stack of the interrupted process is generally used.

Which of the following options is/are True?

[✓ A. S1]

[✓ B. S2]

[✓ C. S1 and S2]

D. None of the above

Answer : A, ~~and~~ B and C .

S1 : True

S2 : True

(Q) Consider the following statements S1 and S2 :

S1 : In the absence of system calls, a user process can never dynamically allocate memory

S2 : The wait() system call inside the parent process returns the pid of an exiting child process .

The number of incorrect statements is / are $\underline{\ 0\ }$

(Q) (MCQ)

Which of the following is not a system call?

A. fork()

B. exec()

C. kill()

[✓ D.] None of the above