A program/command when executed, a special instance is provided by the system to the process. This instance consists of all the services/resources that may be utilized by the process under execution.

* 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 called 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.
* At any point of time, no two processes with the same pid exist in the system because it is the pid that Unix uses to track each process.

**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/geeksforgeeks/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.

**Tracking ongoing processes**

ps (Process status) can be used to see/list all the running processes.

**$ ps**

PID TTY TIME CMD

19 pts/1 00:00:00 sh

24 pts/1 00:00:00 ps

For more information -f (full) can be used along with ps

**$ ps –f**

UID PID PPID C STIME TTY TIME CMD

52471 19 1 0 07:20 pts/1 00:00:00f sh

52471 25 19 0 08:04 pts/1 00:00:00 ps -f

For single-process information, ps along with process id is used

**$ ps 19**

PID TTY TIME CMD

19 pts/1 00:00:00 sh

For a running program (named process) **Pidof** finds the process id’s (pids)   
**Fields described by ps are described as:**

* **UID**: User ID that this process belongs to (the person running it)
* **PID**: Process ID
* **PPID**: Parent process ID (the ID of the process that started it)
* **C**: CPU utilization of process
* **STIME**: Process start time
* **TTY**: Terminal type associated with the process
* **TIME**: CPU time is taken by the process
* **CMD**: The command that started this process

**There are other options which can be used along with ps command :**

* **-a**: Shows information about all users
* **-x**: Shows information about processes without terminals
* **-u**: Shows additional information like -f option
* **-e**: Displays extended information

**Stopping a process:**  
When running in foreground, hitting Ctrl + c (interrupt character) will exit the command. For processes running in background kill command can be used if it’s pid is known.

**$ ps –f**

UID PID PPID C STIME TTY TIME CMD

52471 19 1 0 07:20 pts/1 00:00:00 sh

52471 25 19 0 08:04 pts/1 00:00:00 ps –f

**$ kill 19**

Terminated

If a process ignores a regular kill command, you can use kill -9 followed by the process ID.

**$ kill -9 19**

Terminated

**Other process commands:**

**bg**: A job control command that resumes suspended jobs while keeping them running in the background   
Syntax:

**bg [ job ]**

For example:

bg %19

**fg**: It continues a stopped job by running it in the foreground.   
Syntax:

**fg [ %job\_id ]**

For example

fg 19

**top**: This command is used to show all the running processes within the working environment of Linux.   
Syntax:

top

**nice**: It starts a new process (job) and assigns it a priority (nice) value at the same time.   
Syntax:

nice [-nice value]

nice value ranges from -20 to 19, where -20 is of the highest priority.

**renice** : To change the priority of an already running process renice is used.   
Syntax:

renice [-nice value] [process id]

**df**: It shows the amount of available disk space being used by file systems   
Syntax:

df

Output:

Filesystem 1K-blocks Used Available Use% Mounted on

/dev/loop0 18761008 15246876 2554440 86% /

none 4 0 4 0% /sys/fs/cgroup

udev 493812 4 493808 1% /dev

tmpfs 100672 1364 99308 2% /run

none 5120 0 5120 0% /run/lock

none 503352 1764 501588 1% /run/shm

none 102400 20 102380 1% /run/user

/dev/sda3 174766076 164417964 10348112 95% /host

**free**: It shows the total amount of free and used physical and swap memory in the system, as well as the buffers used by the kernel   
Syntax:

free

Output:

total used free shared buffers cached

Mem: 1006708 935872 70836 0 148244 346656

-/+ buffers/cache: 440972 565736

Swap: 262140 130084 132056

**Types of Processes**

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. But at times when the parent process is killed before the termination of the child process, the child processes become orphan processes, with the parent of all processes “init” process, becomes their new pid.   
   A process which is killed but still shows its entry in the process status or the process table is called a zombie process, they are dead and are not used.
3. **Daemon process :**They are system-related background processes that often run with the permissions of root and services requests from other processes, they most of the time run in the background and wait for processes it can work along with for ex print daemon.   
   When ps –ef is executed, the process with ? in the tty field are daemon processes.