Process Management in Linux

A process means program in execution. There are basically 2 types of processes.

1) Foreground processes :- Such kind of processes are also known as **interactive processes**. These are the processes which are to be executed or initiated by the user or the programmer, they can not be initialized by system services. Such processes take input from the user and return the output. While these processes are running we can not directly initiate a new process from the same terminal.

2) Background processes: Such kind of processes are also known as **non interactive processes**. These are the processes that are to be executed or initiated by the system itself or by users, though they can even be managed by users. These processes have a unique PID or process if assigned to them and we can initiate other processes within the same terminal from which they are initiated.

Foreground Process :-

1) To stop a process in between of its execution , use **sleep command** . To force stop a foreground process in between of its execution - ***CTRL+Z***

sleep NUMBER[SUFFIX]... (by default time in seconds)

suffix may be (s - seconds , m – minutes , h - hours , d - days)

2) To get the list of jobs that are either running or stopped - **jobs command** is used .

It will display the stopped processes in this terminal and even the pending ones.

JOB Job name or number.

-l Lists process IDs in addition to the normal information.

-n List only processes that have changed status since the last notification.

-p Lists process IDs only.

-r Restrict output to running jobs.

-s Restrict output to stopped jobs.

a) To display the process ID or jobs for the job whose name begins with “p,”

```
$ jobs -p %p      OR         $ jobs %p
```

O/p

```
[4]- Stopped      ping cybercity.iz
```

b) Only **-p option to jobs command used to display PIDs only**

```
$ jobs -p
```

O/p

```
7897
7905
7950
8046
```

c) Only **-r option to jobs command used to display only running jobs**

```
$ jobs –r
```

O/p

```
[1]  Running         gpass &
[2]  Running         gnome-calculator &
[3] -Running         gedit fetch-stock-prices.py &
```

3) To run all the pending and force stopped jobs in the background **bg command** is used. This will start the stopped and pending processes in the background.

4) To get details of a process running in background.

```
ps -ef  | grep   name of process or pattern to search
```

ps - list processes

-e - show all processes, not just those belonging to the user

-f - show processes in full format (more detailed than default)

command 1 | command 2 - pass output of command 1 as input to command 2

grep find lines containing a pattern

5) To run processes with priority.

nice -n 5 **process name**

The top priority is -20 but as it may affect the system processes so we have used the priority 5.

Background Processes

A background process runs without being connected to your keyboard. If the background process requires any keyboard input, it waits.

The advantage of running a process in the background is that you can run other commands; you do not have to wait until it completes to start another!

The simplest way to start a background process is to add an ampersand (**&**) at the end of the command.

There are five types of Process in Linux

1. **Parent process :-** The process created by the user on the terminal. All processes have a parent process, If it was created directly by user then the parent process will be the kernel process.

2. **Child process :-** The process created by another process (by its parent process). All child processes have a parent process.

3. **Orphan process :-** Sometimes when the parent gets executed before its own child process then the child process becomes an orphan process. The orphan process have "Init" process (PID 0) as their PPID (parent process ID)

4. **Zombie process :-** The processes which are already dead but shows up in process status is called Zombie process. Zombie processes have Zero CPU consumption.

   When a process ends the execution, then it will have an exit status to report to its master process. Because of that little bit of information, the process will remain in the OS process table as a zombie process, which indicates that it is not to be scheduled for future, but this process cannot be completely removed or the process ID will not be used until the exit has been determined and no longer needed.

When a child completes the process, the master process will receive a SIGCHLD signal to indicate that one of its child process has finished the executing; the parent process will typically call the wait() system status at this point. That status will provide the parent with the child's process exit status, and will cause the child process to be removed from the process table.

5. Daemon process :- These are system-related processes that run in the background. A daemon is a program with a unique purpose. They are utility programs that run silently in the background to monitor and take care of certain subsystems to ensure that the operating system runs properly.