# Experiment No- 04

**Aim: Execution of Process Management Commands**

| | |
|---|---|
| Roll No. | 17 |
| Name | Manav Jawrani |
| Class | D10A |
| Subject | Unix Lab |
| Lab Outcome | LO4: To understand process management and memory management commands in Unix. |
| Date of Performance/ Submission | 16/2/2022-23/2/2022 |

**Aim**: Execution of Process Management Commands of UNIX.

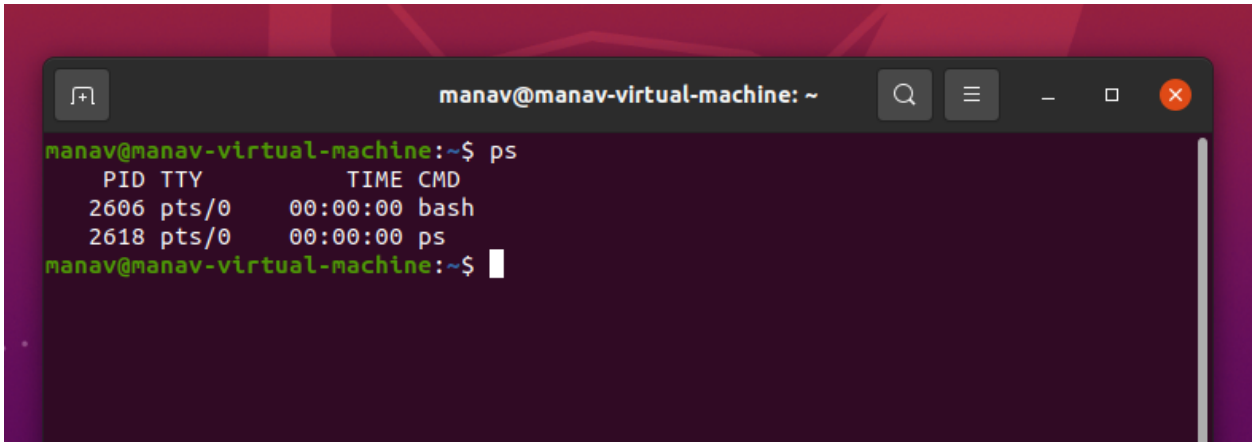
**Introduction**:

A process means program in execution. It generally takes an input, processes it and gives us the appropriate output.There are basically 2 types of processes.

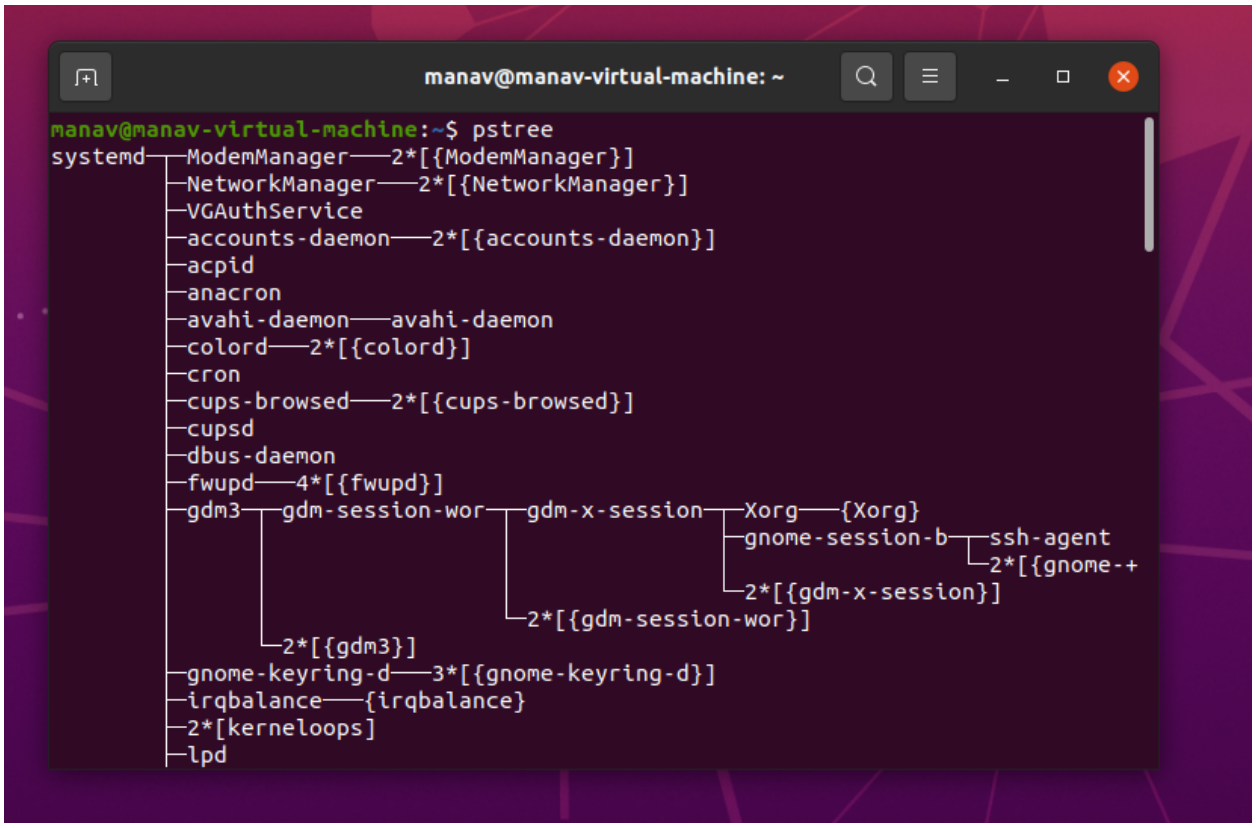
1. **Foreground Processes** - By default, every process that you start runs in the foreground. It gets its input from the keyboard and sends its output to the screen.The process runs in the foreground, the output is directed to my screen, and if the ls command wants any input (which it does not), it waits for it from the keyboard.While a program is running in the foreground and is time-consuming, no other commands can be run (start any other processes) because the prompt would not be available until the program finishes processing and comes out.
2. **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!

**Theory:**

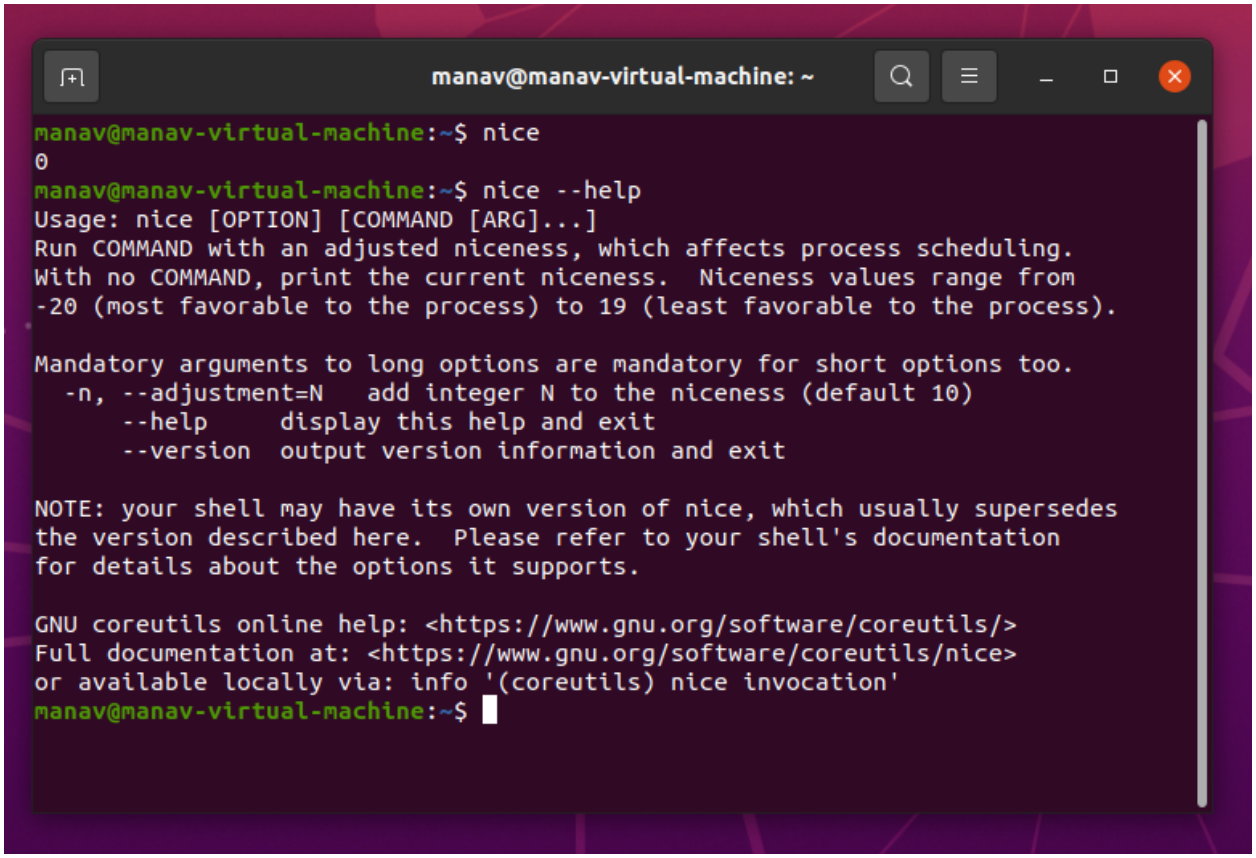
A. **ps -** The abbreviation for ps is "Process Status". ps command is used to list the currently running processes and their PIDs along with some other information depending on different options. It reads the process information from the virtual files in /proc file-system.



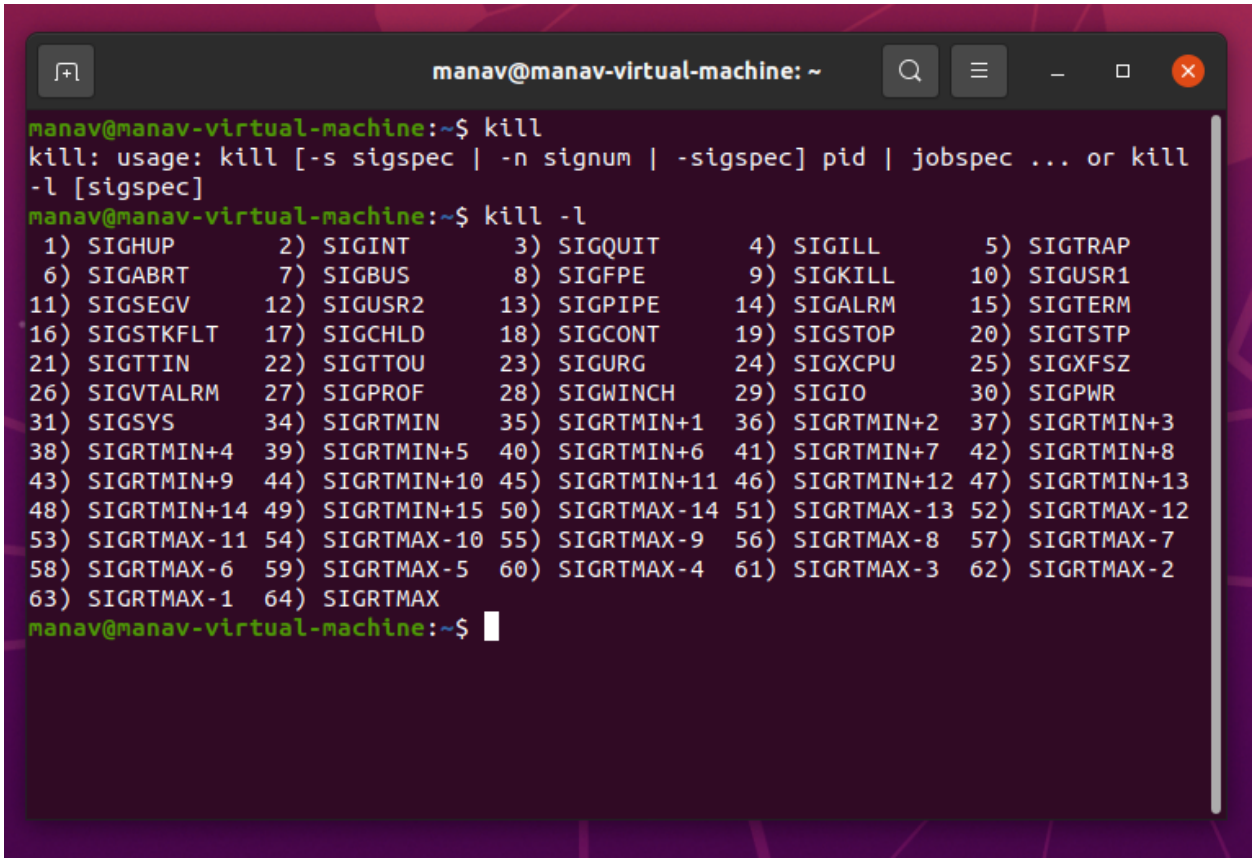
**B. pstree -** Pstree command shows the running processes as a tree which is a more convenient way to display the processes hierarchy and makes the output more visually appealing. The root of the tree is either init or the process with the given pid.



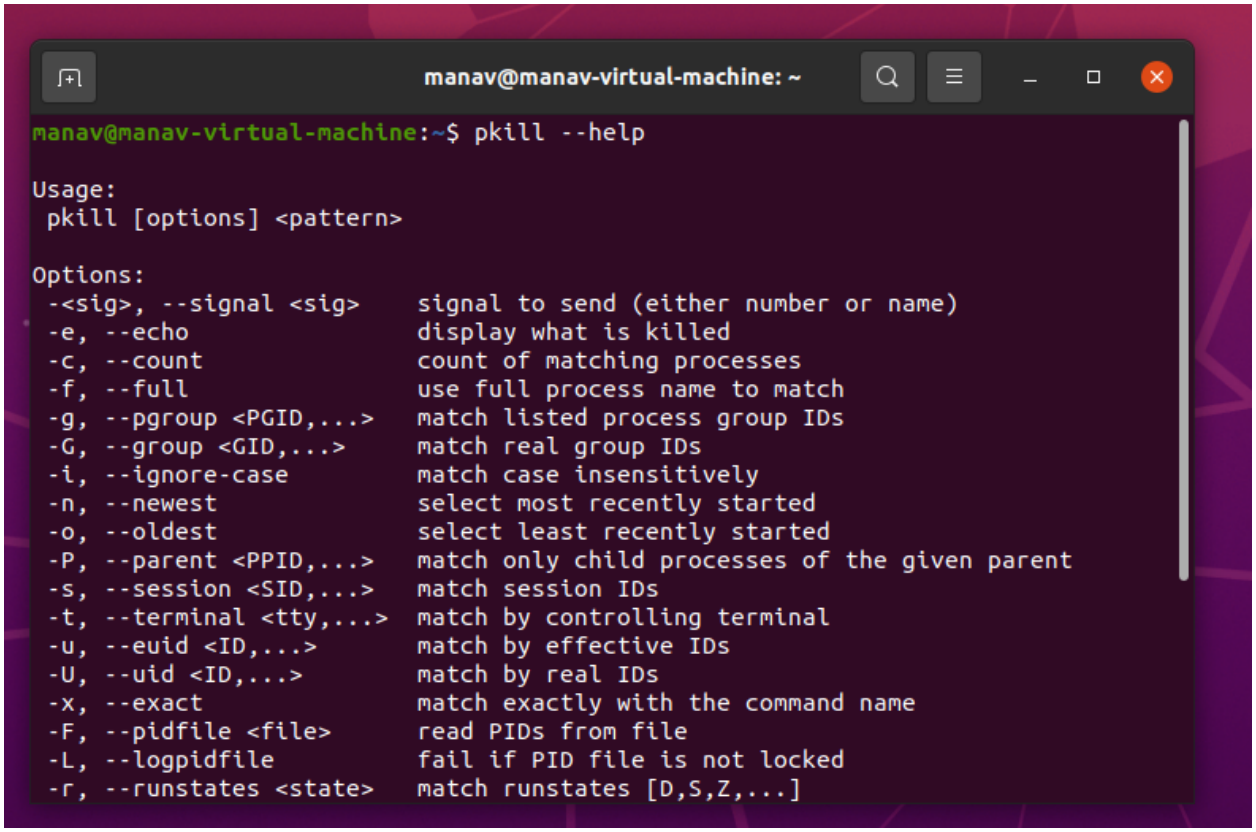

**C. nice -** nice command in Linux helps in execution of a program/process with modified scheduling priority. It launches a process with a user-defined scheduling priority. In this, if we give a process a higher priority, then Kernel will allocate more CPU time to that process.




**D. kill -** kill command in Linux (located in /bin/kill), is a built-in command which is used to terminate processes manually. kill command sends a signal to a process which terminates the process. If the user doesn't specify any signal which is to be sent along with the kill command then a default TERM signal is sent that terminates the process.




E. **pkill -** pkill is a command-line utility that sends signals to the processes of a running program based on given criteria. The processes can be specified by their full or partial names, a user running the process, or other attributes.




F. **killall -** The killall command in Linux is a utility command used for killing any running process on the system based on a given name. This command will terminate the processes forcibly when a specified name matches. The easiest way to kill a bunch of processes altogether is through the killall command.

```
manav@manav-virtual-machine: ~
manav@manav-virtual-machine:~$ killall
Usage: killall [ -Z CONTEXT ] [ -u USER ] [ -y TIME ] [ -o TIME ] [ -eIgiqrvw ]
               [ -s SIGNAL | -SIGNAL ] NAME...
       killall -l, --list
       killall -V, --version

  -e,--exact          require exact match for very long names
  -I,--ignore-case    case insensitive process name match
  -g,--process-group  kill process group instead of process
  -y,--younger-than   kill processes younger than TIME
  -o,--older-than     kill processes older than TIME
  -i,--interactive    ask for confirmation before killing
  -l,--list           list all known signal names
  -q,--quiet          don't print complaints
  -r,--regexp         interpret NAME as an extended regular expression
  -s,--signal SIGNAL  send this signal instead of SIGTERM
  -u,--user USER      kill only process(es) running as USER
  -v,--verbose        report if the signal was successfully sent
  -V,--version        display version information
  -w,--wait           wait for processes to die
  -n,--ns PID         match processes that belong to the same namespaces
                      as PID
  -Z,--context REGEXP kill only process(es) having context
                      (must precede other arguments)
```

**G. xkill -** Command xkill is used to kill a process on X server without passing process name or PID. It forces the X server to close the communication with its clients, which ultimately kills its clients by its X resource. In short, xkill instructs X server to terminate the client.

```
manav@manav-virtual-machine: ~
manav@manav-virtual-machine:~$ xkill -version
xkill 1.0.5
manav@manav-virtual-machine:~$
```

**H. fg -** The fg command, short for the foreground, is a command that moves the background process on your current shell to the foreground.

```
manav@manav-virtual-machine: ~
manav@manav-virtual-machine:~$ fg
bash: fg: current: no such job
manav@manav-virtual-machine:~$ fg --help
fg: fg [job_spec]
    Move job to the foreground.

    Place the job identified by JOB_SPEC in the foreground, making it the
    current job.  If JOB_SPEC is not present, the shell's notion of the
    current job is used.

    Exit Status:
    Status of command placed in foreground, or failure if an error occurs.
manav@manav-virtual-machine:~$
```

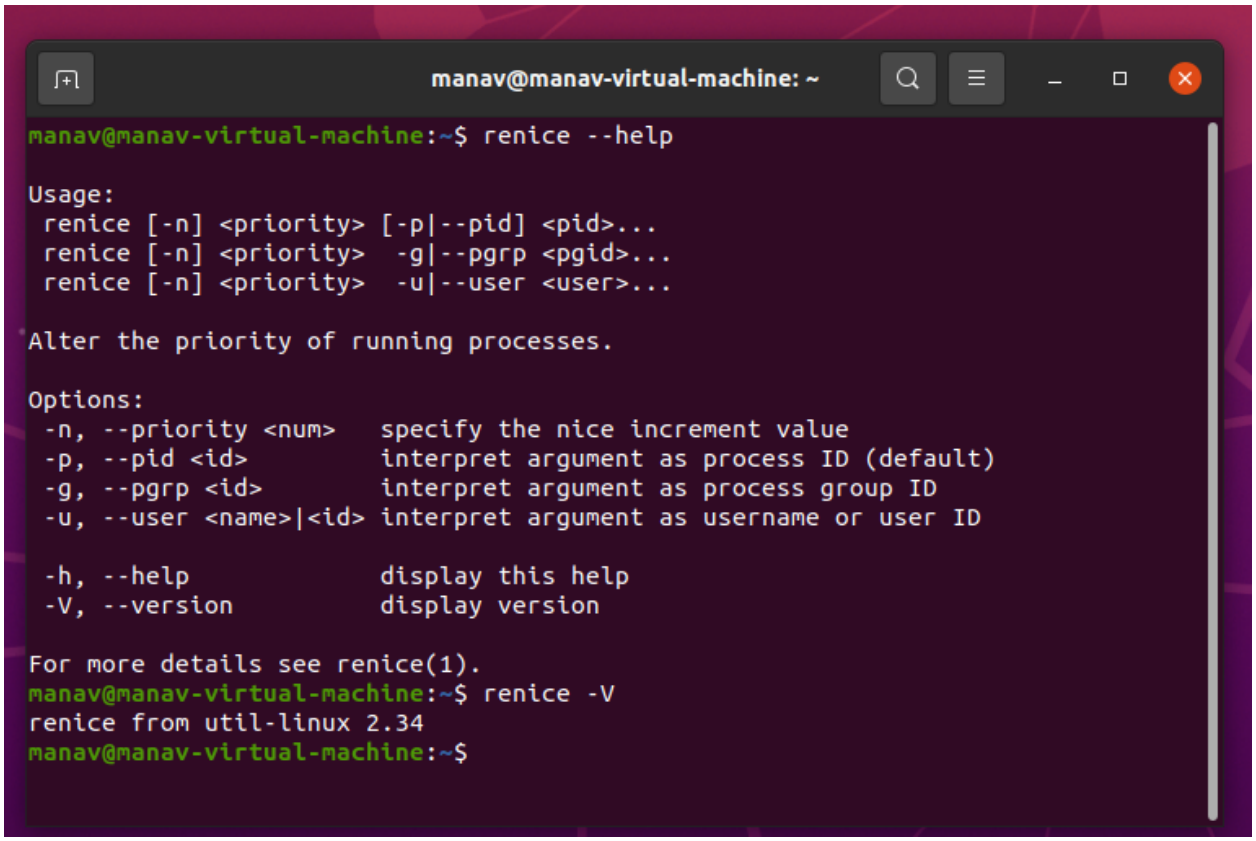
I. **bg -** The bg command, short for the background, is a command that moves the foreground process on your current shell to the background.

```
manav@manav-virtual-machine: ~
manav@manav-virtual-machine:~$ bg
bash: bg: current: no such job
manav@manav-virtual-machine:~$ bg --help
bg: bg [job_spec ...]
    Move jobs to the background.

    Place the jobs identified by each JOB_SPEC in the background, as if they
    had been started with `&'.  If JOB_SPEC is not present, the shell's notion
    of the current job is used.

    Exit Status:
    Returns success unless job control is not enabled or an error occurs.
manav@manav-virtual-machine:~$
```

J. **pgrep -** pgrep is a command-line utility that allows you to find the process IDs of a running program based on given criteria. It can be a full or partial process name, a user running the process, or other attributes.

```
manav@manav-virtual-machine: ~
manav@manav-virtual-machine:~$ pgrep ssh
1728
manav@manav-virtual-machine:~$ pgrep ssh -l
1728 ssh-agent
manav@manav-virtual-machine:~$
```

**K. renice -** The renice command alters the nice value of one or more running processes.




**<u>Conclusion</u>**:

We have understood and executed the process management commands of UNIX.