

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.

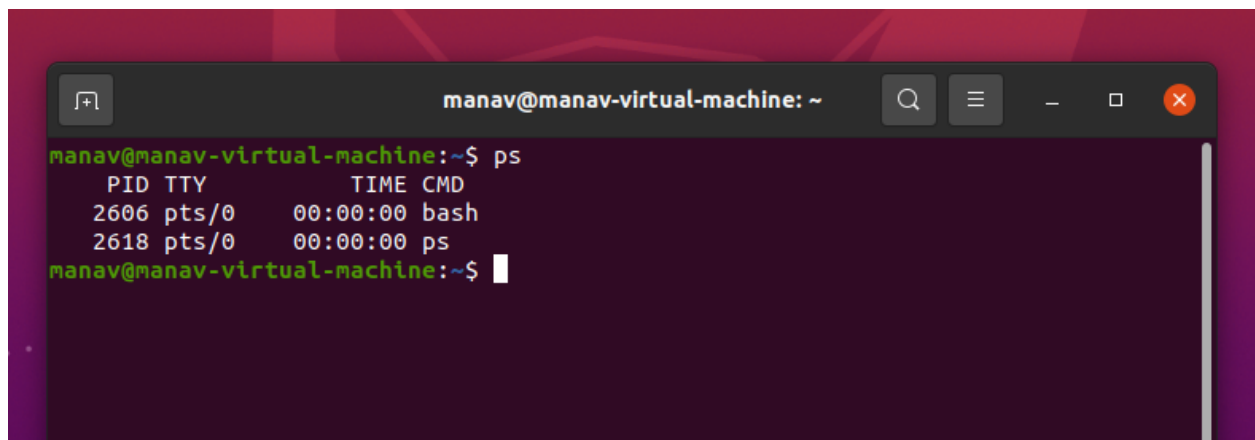
Introduction:

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 process 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!

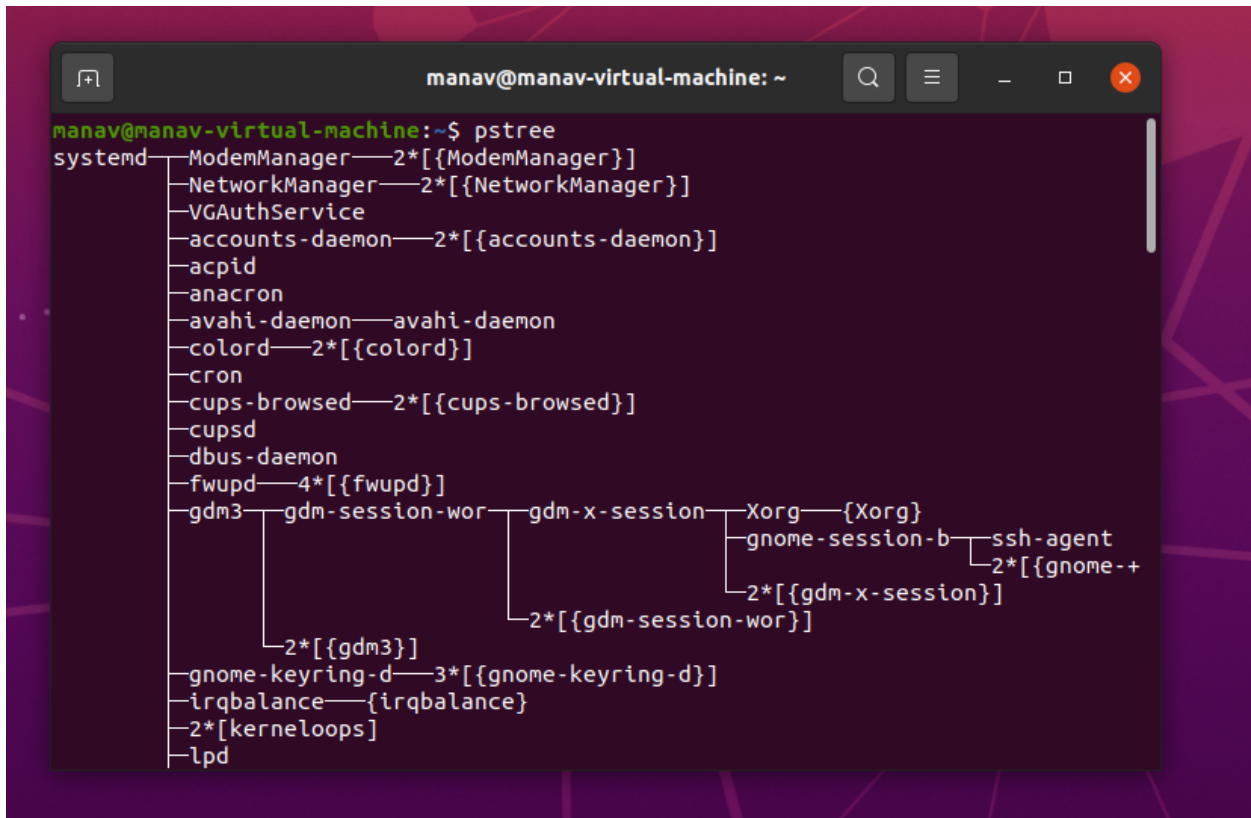
Theory:

- 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.

A screenshot of a terminal window titled "manav@manav-virtual-machine: ~". The terminal shows the command "ps" being executed, which displays a table of running processes. The table has four columns: PID, TTY, TIME, and CMD. There are two processes listed: PID 2606, TTY pts/0, TIME 00:00:00, CMD bash; and PID 2618, TTY pts/0, TIME 00:00:00, CMD ps. The prompt "manav@manav-virtual-machine:~\$" is visible at the bottom.

```
manav@manav-virtual-machine:~$ ps
  PID TTY          TIME CMD
 2606 pts/0    00:00:00 bash
 2618 pts/0    00:00:00 ps
manav@manav-virtual-machine:~$
```

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.



```
manav@manav-virtual-machine:~$ pstree
systemd--ModemManager--2*[{ModemManager}]
        |NetworkManager--2*[{NetworkManager}]
        |VGAuthService
        |accounts-daemon--2*[{accounts-daemon}]
        |acpid
        |anacron
        |avahi-daemon--avahi-daemon
        |colord--2*[{colord}]
        |cron
        |cups-browsed--2*[{cups-browsed}]
        |cupsd
        |dbus-daemon
        |fwupd--4*[{fwupd}]
        |gdm3--gdm-session-wor--gdm-x-session--Xorg--{Xorg}
        |                |                |gnome-session-b--ssh-agent
        |                |                |                |2*[{gnome-+
        |                |                |                |2*[{gdm-x-session}]
        |                |                |2*[{gdm-session-wor}]
        |                |2*[{gdm3}]
        |gnome-keyring-d--3*[{gnome-keyring-d}]
        |irqbalance--{irqbalance}
        |2*[kerneleofs]
        |lpd
```

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.

A terminal window titled 'manav@manav-virtual-machine: ~' with standard window controls. The terminal shows the execution of the 'nice' command and its help output. The prompt is 'manav@manav-virtual-machine:~\$'. The first command is 'nice', which outputs '0'. The second command is 'nice --help', which displays the usage and options for the 'nice' command. The output includes a usage line, a description of the command, a range of niceness values, mandatory arguments for long options, and a list of short options with their descriptions. It also includes a note about the shell's version and links to GNU coreutils online help and full documentation.

```
manav@manav-virtual-machine:~$ nice
0
manav@manav-virtual-machine:~$ nice --help
Usage: nice [OPTION] [COMMAND [ARG]...]
Run COMMAND with an adjusted niceness, which affects process scheduling.
With no COMMAND, print the current niceness.  Niceness values range from
-20 (most favorable to the process) to 19 (least favorable to the process).

Mandatory arguments to long options are mandatory for short options too.
  -n, --adjustment=N  add integer N to the niceness (default 10)
  --help              display this help and exit
  --version            output version information and exit

NOTE: your shell may have its own version of nice, which usually supersedes
the version described here.  Please refer to your shell's documentation
for details about the options it supports.

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Full documentation at: <https://www.gnu.org/software/coreutils/nice>
or available locally via: info '(coreutils) nice invocation'
manav@manav-virtual-machine:~$
```

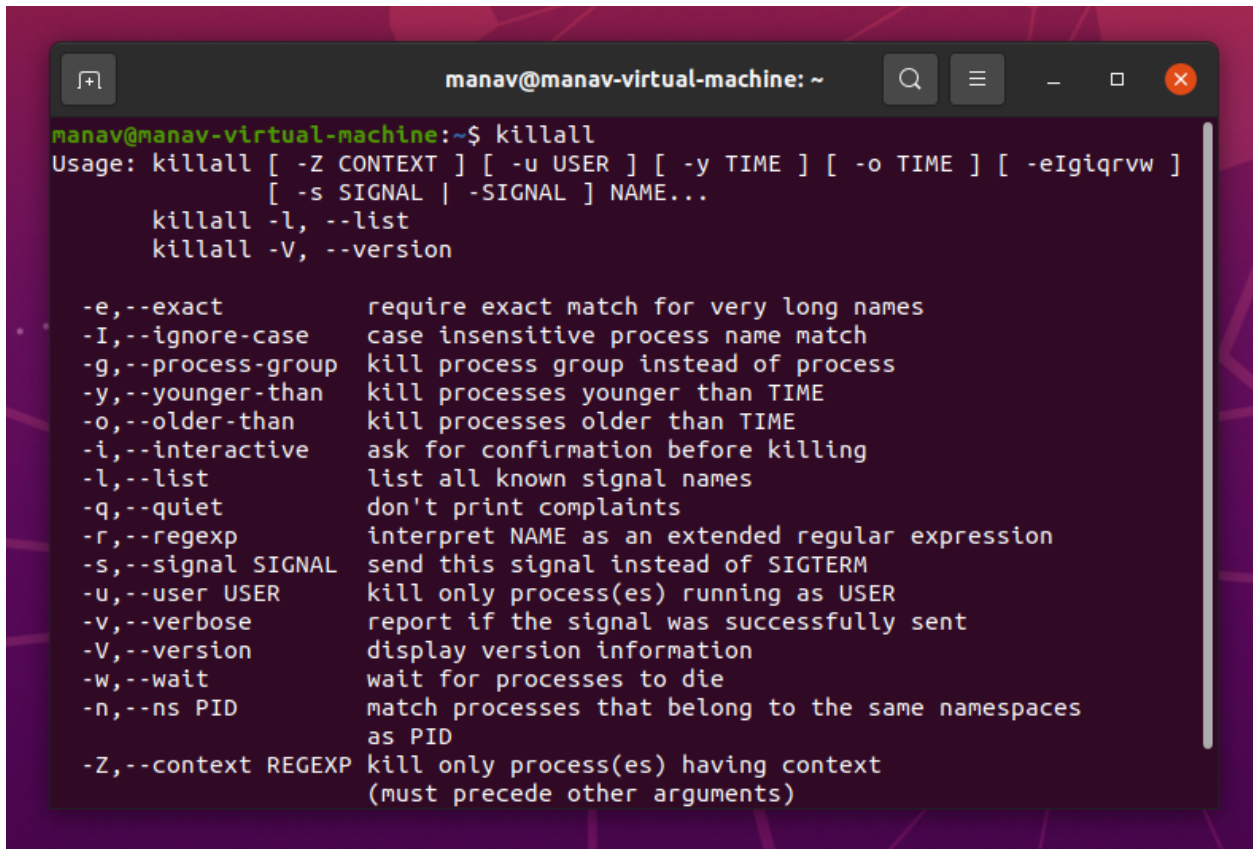
D. kill - kill command in Linux (located in /bin/kill), is a built-in command which is used to terminate processes manually. kill command sends a signal to a process which terminates the process. If the user doesn't specify any signal which is to be sent along with the kill command then a default TERM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

```
manav@manav-virtual-machine: ~  
manav@manav-virtual-machine:~$ kill  
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill  
-l [sigspec]  
manav@manav-virtual-machine:~$ kill -l  
1) SIGHUP      2) SIGINT      3) SIGQUIT     4) SIGILL      5) SIGTRAP  
6) SIGABRT     7) SIGBUS     8) SIGFPE      9) SIGKILL     10) SIGUSR1  
11) SIGSEGV    12) SIGUSR2    13) SIGPIPE    14) SIGALRM     15) SIGTERM  
16) SIGSTKFLT  17) SIGCHLD   18) SIGCONT    19) SIGSTOP     20) SIGTSTP  
21) SIGTTIN    22) SIGTTOU    23) SIGURG     24) SIGXCPU     25) SIGXFSZ  
26) SIGVTALRM  27) SIGPROF   28) SIGWINCH   29) SIGIO       30) SIGPWR  
31) SIGSYS     34) SIGRTMIN   35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3  
38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8  
43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13  
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12  
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7  
58) SIGRTMAX-6 59) SIGRTMAX-5 60) SIGRTMAX-4 61) SIGRTMAX-3 62) SIGRTMAX-2  
63) SIGRTMAX-1 64) SIGRTMAX  
manav@manav-virtual-machine:~$
```

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.

```
manav@manav-virtual-machine: ~  
manav@manav-virtual-machine:~$ pkill --help  
  
Usage:  
pkill [options] <pattern>  
  
Options:  
-<sig>, --signal <sig>    signal to send (either number or name)  
-e, --echo                display what is killed  
-c, --count                count of matching processes  
-f, --full                use full process name to match  
-g, --pgroup <PGID,...>  match listed process group IDs  
-G, --group <GID,...>    match real group IDs  
-i, --ignore-case          match case insensitively  
-n, --newest               select most recently started  
-o, --oldest               select least recently started  
-P, --parent <PPID,...>  match only child processes of the given parent  
-s, --session <SID,...>  match session IDs  
-t, --terminal <tty,...> match by controlling terminal  
-u, --euid <ID,...>       match by effective IDs  
-U, --uid <ID,...>        match by real IDs  
-x, --exact                match exactly with the command name  
-F, --pidfile <file>      read PIDs from file  
-L, --logpidfile           fail if PID file is not locked  
-r, --runstates <state>  match runstates [D,S,Z,...]
```

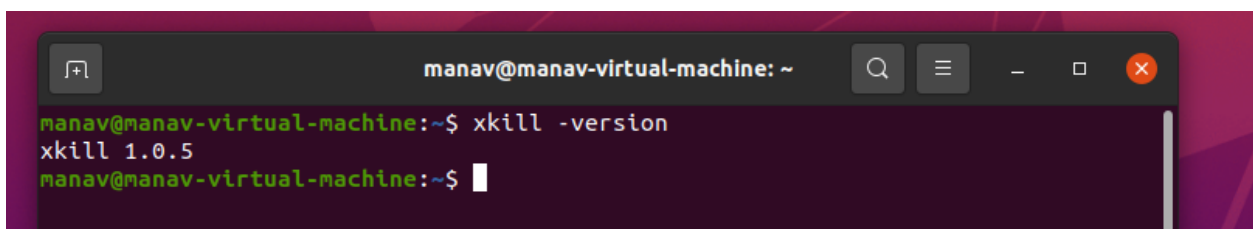
F. killall - The killall command in Linux is a utility command used for killing any running process on the system based on a given name. This command will terminate the processes forcibly when a specified name matches. The easiest way to kill a bunch of processes altogether is through the killall command.

A terminal window titled 'manav@manav-virtual-machine: ~' showing the output of the 'killall' command. The output displays the usage of 'killall' with various options and their descriptions. The options listed are: -e, --exact; -I, --ignore-case; -g, --process-group; -y, --younger-than; -o, --older-than; -i, --interactive; -l, --list; -q, --quiet; -r, --regex; -s, --signal SIGNAL; -u, --user USER; -v, --verbose; -V, --version; -w, --wait; -n, --ns PID; and -Z, --context REGEXP. Each option is followed by a brief description of its function.

```
manav@manav-virtual-machine:~$ killall
Usage: killall [ -Z CONTEXT ] [ -u USER ] [ -y TIME ] [ -o TIME ] [ -eIgiqrwv ]
        [ -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,--regex          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.

A terminal window titled 'manav@manav-virtual-machine: ~' showing the output of the 'xkill -version' command. The output is 'xkill 1.0.5'.

```
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:~$
```

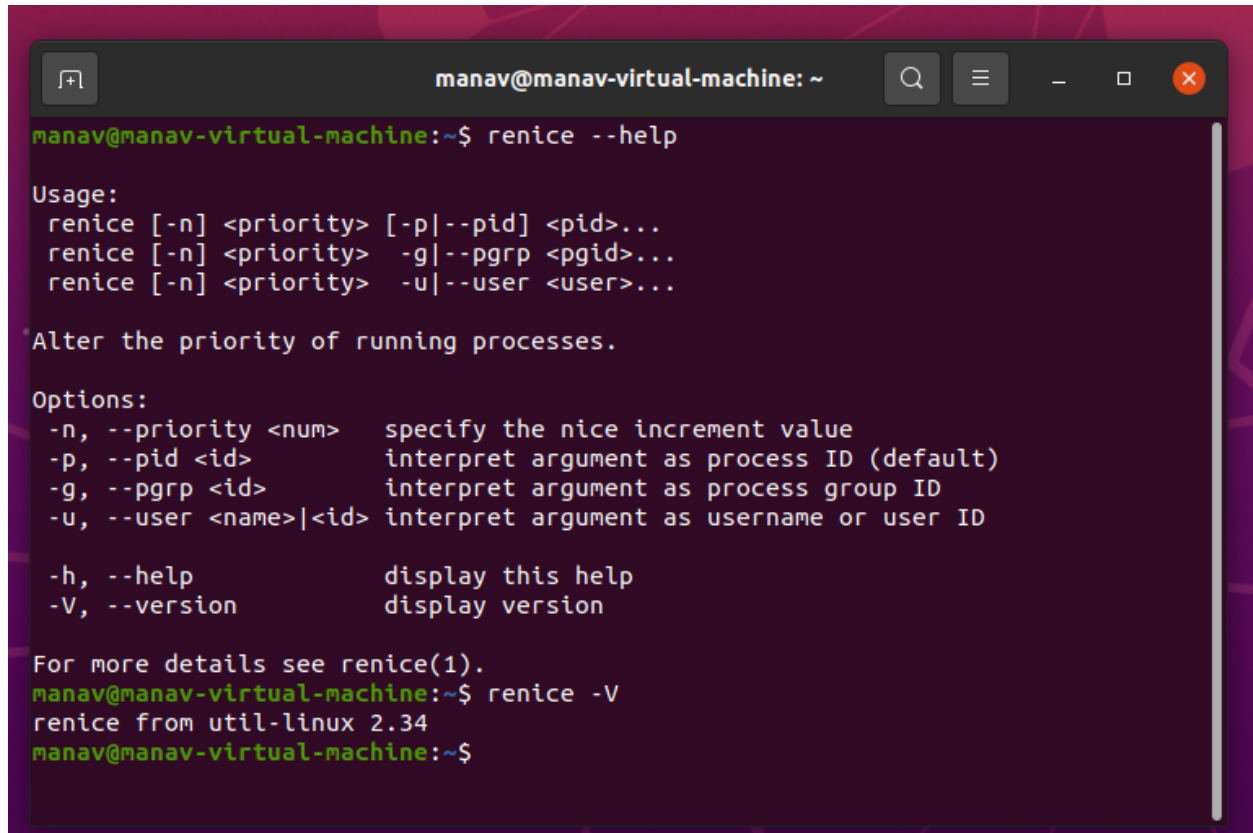
- I. **bg** - The bg command, short for the background, is a command that moves the foreground process on your current shell to the background.

```
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.

A terminal window titled 'manav@manav-virtual-machine: ~' with standard window controls. The terminal shows the command 'renice --help' and its output. The output includes usage instructions, a description of the command's purpose, a list of options with their descriptions, and the version information. The prompt is green, and the command and its output are white on a dark background.

```
manav@manav-virtual-machine:~$ renice --help

Usage:
renice [-n] <priority> [-p|--pid] <pid>...
renice [-n] <priority> -g|--pgrp <pgid>...
renice [-n] <priority> -u|--user <user>...

Alter the priority of running processes.

Options:
-n, --priority <num>    specify the nice increment value
-p, --pid <id>          interpret argument as process ID (default)
-g, --pgrp <id>         interpret argument as process group ID
-u, --user <name>|<id>  interpret argument as username or user ID

-h, --help              display this help
-V, --version           display version

For more details see renice(1).
manav@manav-virtual-machine:~$ renice -V
renice from util-linux 2.34
manav@manav-virtual-machine:~$
```

Conclusion:

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