The ps Command. Linux provides a utility called ps (which stands as abbreviation for “Process Status”) for **viewing information related with the processes on a system**. The ps command is used to list the currently running processes and their PIDs along with some other information, which depends on different options.

**Syntax**:

**ps [ OPTIONS ]**

**Command Options:**

| **Option** | **Meaning** |
| --- | --- |
| -a | Write information for all processes associated with terminals. Implementations may omit session leaders from this list. |
| -A | Write information for all processes. |
| -d | Write information for all processes, except session leaders. |
| -e | Write information for all processes. (Equivalent to -A.) |
| -f | Generate a full listing. (See the STDOUT section for the contents of a full listing.) |
| -g grouplist | Write information for processes whose session leaders are given in grouplist. The application shall ensure that the grouplist is a single argument in the form of a <blank> or <comma>-separated list. |
| -G grouplist | Write information for processes whose real group ID numbers are given in grouplist. The application shall ensure that the grouplist is a single argument in the form of a <blank> or <comma>-separated list. |
| -l | Generate a long listing. (See STDOUT for the contents of a long listing.) |
| -n namelist | Specify the name of an alternative system namelist file in place of the default. The name of the default file and the format of a namelist file are unspecified. |
| -o format | Write information according to the format specification given in format. This is fully described in the STDOUT section. Multiple -o options can be specified; the format specification shall be interpreted as the <space>-separated concatenation of all the format option-arguments. |
| -p proclist | Write information for processes whose process ID numbers are given in proclist. The application shall ensure that the proclist is a single argument in the form of a <blank> or <comma>-separated list. |
| -t termlist | Write information for processes associated with terminals given in termlist. The application shall ensure that the termlist is a single argument in the form of a <blank> or <comma>-separated list. Terminal identifiers shall be given in an implementation-defined format. On XSI-conformant systems, they shall be given in one of two forms: the device's filename (for example, tty04) or, if the device's filename starts with tty, just the identifier following the characters tty (for example, "04"). |
| -u userlist | Write information for processes whose user ID numbers or login names are given in userlist. The application shall ensure that the userlist is a single argument in the form of a <blank> or <comma>-separated list. In the listing, the numerical user ID shall be written unless the -f option is used, in which case the login name shall be written. |
| -U userlist | Write information for processes whose real user ID numbers or login names are given in userlist. The application shall ensure that the userlist is a single argument in the form of a <blank> or <comma>-separated list. |

pbmac@pbmac-server $ ps

PID TTY TIME CMD

11845 pts/1 00:00:00 bash

17636 pts/1 00:00:00 ps

The result contains four columns of information:  
PID – the unique process ID.  
TTY – terminal type that the user running this command is logged into.  
TIME – amount of CPU in minutes and seconds that the process has been running.  
CMD – name of the command that launched the process.

**Note –** Sometimes when we execute **ps** command, it shows TIME as 00:00:00. It is nothing but the total accumulated CPU utilization time for any process and 00:00:00 indicates no CPU time has been given by the kernel till now. In the above example we found that for bash no CPU time has been given. This is because bash is just a parent process for different processes which needs bash for their execution, and bash itself is not utilizing any CPU time till now.

There are numerous options for the **ps** command - and the output is different with the different options.

pbmac-server $ ps -ax

PID TTY STAT TIME COMMAND

1 ? Ss 0:27 /sbin/init

2 ? S 0:00 [kthreadd]

6 ? I< 0:00 [mm\_percpu\_wq]

7 ? S 0:07 [ksoftirqd/0]

Notice the TTY is a question mark, because this process is NOT associated with a terminal session. The STAT column refers to what state (as previously discussed) the process is in.

**pbmac**@pbmac-server $ ps -ef

UID PID PPID C STIME TTY TIME CMD

root 1 0 0 Sep09 ? 00:00:27 /sbin/init

root 2 0 0 Sep09 ? 00:00:00 [kthreadd]

root 6 2 0 Sep09 ? 00:00:00 [mm\_percpu\_wq]

root 7 2 0 Sep09 ? 00:00:07 [ksoftirqd/0]

Using the -ef options gives a slightly different output:

UID - the user who owns this process  
PPID - the parent process ID - when the PPID is 0 that means the process was started by the boot process.

There are quite a number of options - the **ps -ax** and **ps -ef** commands shown above are quite common. You are encouraged to view the **ps** man page and familiarize yourself with some of the other options, as there are just too many to cover here.

The **ps** command accepts three styles of options:

1. UNIX options, which may be grouped and must be preceded by a dash.
2. BSD options, which may be grouped and must not be used with a dash.
3. GNU long options, which are preceded by two dashes.

Options of different types may be freely mixed, but conflicts can appear. There are some synonymous options, which are functionally identical, due to the many standards and **ps** implementations that this **ps** is compatible with.

Note that **ps -aux** is distinct from **ps aux**. The POSIX and UNIX standards require that **ps -aux** print all processes owned by a user named **x**, as well as printing all processes that would be selected by the **-a** option. If the user named **x** does not exist, this **ps** may interpret the command as **ps aux** instead and print a warning. This behavior is intended to aid in transitioning old scripts and habits. It is fragile, subject to change, and thus should not be relied upon.