NAME

ps − report a snapshot of the current processes.

SYNOPSIS

ps [*options*]

DESCRIPTION

ps displays information about a selection of the active processes. If you want a repetitive update of the selection and the displayed information, use **top** instead.

This version of **ps** accepts several kinds 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.

By default, **ps** selects all processes with the same effective user ID (euid=EUID) as the current user and associated with the same terminal as the invoker. It displays the process ID (pid=PID), the terminal associated with the process (tname=TTY), the cumulated CPU time in [DD−]hh:mm:ss format (time=TIME), and the executable name (ucmd=CMD). Output is unsorted by default.

The use of BSD−style options will add process state (stat=STAT) to the default display and show the command args (args=COMMAND) instead of the executable name. You can override this with the **PS_FORMAT** environment variable. The use of BSD−style options will also change the process selection to include processes on other terminals (TTYs) that are owned by you; alternately, this may be described as setting the selection to be the set of all processes filtered to exclude processes owned by other users or not on a terminal. These effects are not considered when options are described as being "identical" below, so **−M** will be considered identical to **Z** and so on.

Except as described below, process selection options are additive. The default selection is discarded, and then the selected processes are added to the set of processes to be displayed. A process will thus be shown if it meets any of the given selection criteria.

EXAMPLES

To see every process on the system using standard syntax:

ps −e
ps −ef
ps −eF
ps −ely

To see every process on the system using BSD syntax:

ps ax
ps axu

To print a process tree:

ps −ejH
ps axjf

To get info about threads:

ps −eLf
ps axms

To get security info:
ps −eo euser,ruser,suser,fuser,f,comm,label
ps axZ
ps −eM

To see every process running as root (real & effective ID) in user format:
ps −U root −u root u

To see every process with a user−defined format:
ps −eo pid,tid,class,rtprio,ni,pri,psr,pcpu,stat,wchan:14,comm
ps axo stat,euid,ruid,tty,tpgid,sess,pgrp,ppid,pid,pcpu,comm
ps −Ao pid,tt,user,fname,tmout,f,wchan

Print only the process IDs of syslogd:
ps −C syslogd −o pid=

Print only the name of PID 42:
ps −q 42 −o comm=

SIMPLE PROCESS SELECTION

a Lift the BSD−style "only yourself" restriction, which is imposed upon the set of all processes when some BSD−style (without "−") options are used or when the **ps** personality setting is BSD−like. The set of processes selected in this manner is in addition to the set of processes selected by other means. An alternate description is that this option causes **ps** to list all processes with a terminal (tty), or to list all processes when used together with the **x** option.

−A Select all processes. Identical to **−e**.

−a Select all processes except both session leaders (see *getsid*(2)) and processes not associated with a terminal.

−d Select all processes except session leaders.

−−deselect
Select all processes except those that fulfill the specified conditions (negates the selection). Identical to **−N**.

−e Select all processes. Identical to **−A**.

g Really all, even session leaders. This flag is obsolete and may be discontinued in a future release. It is normally implied by the **a** flag, and is only useful when operating in the sunos4 personality.

−N Select all processes except those that fulfill the specified conditions (negates the selection). Identical to **−−deselect**.

T Select all processes associated with this terminal. Identical to the **t** option without any argument.

r Restrict the selection to only running processes.

x Lift the BSD−style "must have a tty" restriction, which is imposed upon the set of all processes when some BSD−style (without "−") options are used or when the **ps** personality setting is BSD−like. The set of processes selected in this manner is in addition to the set of processes selected by other means. An alternate description is that this option causes **ps** to list all processes owned by you (same EUID as **ps**), or to list all processes when used together with the **a** option.

PROCESS SELECTION BY LIST

These options accept a single argument in the form of a blank−separated or comma−separated list. They can be used multiple times. For example: **ps −p "1 2" −p 3,4**

−123 Identical to **−−pid** *123*.

123 Identical to **−−pid** *123*.

−C *cmdlist*
Select by command name. This selects the processes whose executable name is given in *cmdlist*. NOTE: The command name is not the same as the command line. Previous versions of procps and

the kernel truncated this command name to 15 characters. This limitation is no longer present in both. If you depended on matching only 15 characters, you may no longer get a match.

−G *grplist*
Select by real group ID (RGID) or name. This selects the processes whose real group name or ID is in the *grplist* list. The real group ID identifies the group of the user who created the process, see *getgid*(2).

−g *grplist*
Select by session OR by effective group name. Selection by session is specified by many standards, but selection by effective group is the logical behavior that several other operating systems use. This **ps** will select by session when the list is completely numeric (as sessions are). Group ID numbers will work only when some group names are also specified. See the **−s** and **−−group** options.

−−Group *grplist*
Select by real group ID (RGID) or name. Identical to **−G**.

−−group *grplist*
Select by effective group ID (EGID) or name. This selects the processes whose effective group name or ID is in *grplist*. The effective group ID describes the group whose file access permissions are used by the process (see *getegid*(2)). The **−g** option is often an alternative to **−−group**.

p *pidlist*
Select by process ID. Identical to **−p** and **−−pid**.

−p *pidlist*
Select by PID. This selects the processes whose process ID numbers appear in *pidlist*. Identical to **p** and **−−pid**.

−−pid *pidlist*
Select by process ID. Identical to **−p** and **p**.

−−ppid *pidlist*
Select by parent process ID. This selects the processes with a parent process ID in *pidlist*. That is, it selects processes that are children of those listed in *pidlist*.

q *pidlist*
Select by process ID (quick mode). Identical to **−q** and **−−quick−pid**.

−q *pidlist*
Select by PID (quick mode). This selects the processes whose process ID numbers appear in *pidlist*. With this option **ps** reads the necessary info only for the pids listed in the *pidlist* and doesn't apply additional filtering rules. The order of pids is unsorted and preserved. No additional selection options, sorting and forest type listings are allowed in this mode. Identical to **q** and **−−quick−pid**.

−−quick−pid *pidlist*
Select by process ID (quick mode). Identical to **−q** and **q**.

−s *sesslist*
Select by session ID. This selects the processes with a session ID specified in *sesslist*.

−−sid *sesslist*
Select by session ID. Identical to **−s**.

t *ttylist* Select by tty. Nearly identical to **−t** and **−−tty**, but can also be used with an empty *ttylist* to indicate the terminal associated with **ps**. Using the **T** option is considered cleaner than using **t** with an empty *ttylist*.

−t *ttylist*
Select by tty. This selects the processes associated with the terminals given in *ttylist*. Terminals (ttys, or screens for text output) can be specified in several forms: /dev/ttyS1, ttyS1, S1. A plain

"−" may be used to select processes not attached to any terminal.

−−tty *ttylist*
Select by terminal. Identical to **−t** and **t**.

U *userlist*
Select by effective user ID (EUID) or name. This selects the processes whose effective user name or ID is in *userlist*. The effective user ID describes the user whose file access permissions are used by the process (see *geteuid*(2)). Identical to **−u** and **−−user**.

−U *userlist*
Select by real user ID (RUID) or name. It selects the processes whose real user name or ID is in the *userlist* list. The real user ID identifies the user who created the process, see *getuid*(2).

−u *userlist*
Select by effective user ID (EUID) or name. This selects the processes whose effective user name or ID is in *userlist*.

The effective user ID describes the user whose file access permissions are used by the process (see *geteuid*(2)). Identical to **U** and **−−user**.

−−User *userlist*
Select by real user ID (RUID) or name. Identical to **−U**.

−−user *userlist*
Select by effective user ID (EUID) or name. Identical to **−u** and **U**.

OUTPUT FORMAT CONTROL

These options are used to choose the information displayed by **ps**. The output may differ by personality.

−c Show different scheduler information for the **−l** option.

−−context
Display security context format (for SELinux).

−f Do full−format listing. This option can be combined with many other UNIX−style options to add additional columns. It also causes the command arguments to be printed. When used with **−L**, the NLWP (number of threads) and LWP (thread ID) columns will be added. See the **c** option, the format keyword **args**, and the format keyword **comm**.

−F Extra full format. See the **−f** option, which **−F** implies.

−−format *format*
user−defined format. Identical to **−o** and **o**.

j BSD job control format.

−j Jobs format.

l Display BSD long format.

−l Long format. The **−y** option is often useful with this.

−M Add a column of security data. Identical to **Z** (for SELinux).

O *format*
is preloaded **o** (overloaded). The BSD **O** option can act like **−O** (user−defined output format with some common fields predefined) or can be used to specify sort order. Heuristics are used to determine the behavior of this option. To ensure that the desired behavior is obtained (sorting or formatting), specify the option in some other way (e.g. with **−O** or **−−sort**). When used as a formatting option, it is identical to **−O**, with the BSD personality.

−O *format*
Like **−o**, but preloaded with some default columns. Identical to **−o pid,***format***,state,tname,time, command** or **−o pid,***format***,tname,time,cmd**, see **−o** below.

o *format*
Specify user−defined format. Identical to **−o** and **−−format**.

−o *format*
User−defined format. *format* is a single argument in the form of a blank−separated or comma−separated list, which offers a way to specify individual output columns. The recognized keywords are described in the **STANDARD FORMAT SPECIFIERS** section below. Headers may be renamed (**ps −o pid,ruser=RealUser −o comm=Command**) as desired. If all column headers are empty (**ps −o pid= −o comm=**) then the header line will not be output. Column width will increase as needed for wide headers; this may be used to widen up columns such as WCHAN (**ps −o pid,wchan=WIDE−WCHAN−COLUMN −o comm**). Explicit width control (**ps opid, wchan:42,cmd**) is offered too. The behavior of **ps −o pid=X,comm=Y** varies with personality; output may be one column named "X,comm=Y" or two columns named "X" and "Y". Use multiple **−o** options when in doubt. Use the **PS_FORMAT** environment variable to specify a default as desired; DefSysV and DefBSD are macros that may be used to choose the default UNIX or BSD columns.

s Display signal format.

u Display user−oriented format.

v Display virtual memory format.

X Register format.

−y Do not show flags; show rss in place of addr. This option can only be used with **−l**.

Z Add a column of security data. Identical to **−M** (for SELinux).

OUTPUT MODIFIERS

c Show the true command name. This is derived from the name of the executable file, rather than from the argv value. Command arguments and any modifications to them are thus not shown. This option effectively turns the **args** format keyword into the **comm** format keyword; it is useful with the **−f** format option and with the various BSD−style format options, which all normally display the command arguments. See the **−f** option, the format keyword **args**, and the format keyword **comm**.

−−cols *n*
Set screen width.

−−columns *n*
Set screen width.

−−cumulative
Include some dead child process data (as a sum with the parent).

e Show the environment after the command.

f ASCII art process hierarchy (forest).

−−forest
ASCII art process tree.

h No header. (or, one header per screen in the BSD personality). The **h** option is problematic. Standard BSD **ps** uses this option to print a header on each page of output, but older Linux **ps** uses this option to totally disable the header. This version of **ps** follows the Linux usage of not printing the header unless the BSD personality has been selected, in which case it prints a header on each page of output. Regardless of the current personality, you can use the long options **−−headers** and **−−no−headers** to enable printing headers each page or disable headers entirely, respectively.

−H Show process hierarchy (forest).

−−headers
Repeat header lines, one per page of output.

k *spec* Specify sorting order. Sorting syntax is [**+**|**−**]*key*[,[**+**|**−**]*key*[,...]]. Choose a multi−letter key from the **STANDARD FORMAT SPECIFIERS** section. The "+" is optional since default direction is increasing numerical or lexicographic order. Identical to **−−sort**.

Examples:
ps jaxkuid,−ppid,+pid
ps axk comm o comm,args
ps kstart_time −ef

−−lines *n*
Set screen height.

n Numeric output for WCHAN and USER (including all types of UID and GID).

−−no−headers
Print no header line at all. **−−no−heading** is an alias for this option.

O *order*
Sorting order (overloaded). The BSD **O** option can act like **−O** (user−defined output format with some common fields predefined) or can be used to specify sort order. Heuristics are used to determine the behavior of this option. To ensure that the desired behavior is obtained (sorting or formatting), specify the option in some other way (e.g. with **−O** or **−−sort**).

For sorting, obsolete BSD **O** option syntax is **O**[**+**|**−**]*k1*[,[**+**|**−**]*k2*[,...]]. It orders the processes listing according to the multilevel sort specified by the sequence of one−letter short keys *k1*,*k2*, ... described in the **OBSOLETE SORT KEYS** section below. The "+" is currently optional, merely re−iterating the default direction on a key, but may help to distinguish an **O** sort from an **O** format. The "−" reverses direction only on the key it precedes.

−−rows *n*
Set screen height.

S Sum up some information, such as CPU usage, from dead child processes into their parent. This is useful for examining a system where a parent process repeatedly forks off short−lived children to do work.

−−sort *spec*
Specify sorting order. Sorting syntax is [+|−]*key*[,[**+**|**−**]*key*[,...]]. Choose a multi−letter key from the **STANDARD FORMAT SPECIFIERS** section. The "+" is optional since default direction is increasing numerical or lexicographic order. Identical to **k**. For example: **ps jax −−sort=uid,−ppid,+pid**

w Wide output. Use this option twice for unlimited width.

−w Wide output. Use this option twice for unlimited width.

−−width *n*
Set screen width.

THREAD DISPLAY

H Show threads as if they were processes.

−L Show threads, possibly with LWP and NLWP columns.

m Show threads after processes.

−m Show threads after processes.

−T Show threads, possibly with SPID column.

OTHER INFORMATION

−−help *section*
Print a help message. The *section* argument can be one of *s*imple, *l*ist, *o*utput, *t*hreads, *m*isc, or *a*ll. The argument can be shortened to one of the underlined letters as in: s|l|o|t|m|a.

−−info Print debugging info.

L List all format specifiers.

V Print the procps-ng version.

−V Print the procps-ng version.

−−version
Print the procps-ng version.

NOTES

This **ps** works by reading the virtual files in /proc. This **ps** does not need to be setuid kmem or have any privileges to run. Do not give this **ps** any special permissions.

CPU usage is currently expressed as the percentage of time spent running during the entire lifetime of a process. This is not ideal, and it does not conform to the standards that **ps** otherwise conforms to. CPU usage is unlikely to add up to exactly 100%.

The SIZE and RSS fields don't count some parts of a process including the page tables, kernel stack, struct thread_info, and struct task_struct. This is usually at least 20 KiB of memory that is always resident. SIZE is the virtual size of the process (code+data+stack).

Processes marked <defunct> are dead processes (so−called "zombies") that remain because their parent has not destroyed them properly. These processes will be destroyed by *init*(8) if the parent process exits.

If the length of the username is greater than the length of the display column, the username will be truncated. See the **−o** and **−O** formatting options to customize length.

Commands options such as **ps −aux** are not recommended as it is a confusion of two different standards. According to the POSIX and UNIX standards, the above command asks to display all processes with a TTY (generally the commands users are running) plus all processes owned by a user named *x*. If that user doesn't exist, then **ps** will assume you really meant **ps aux**.

PROCESS FLAGS

The sum of these values is displayed in the "F" column, which is provided by the **flags** output specifier:

1 forked but didn't exec
4 used super−user privileges

PROCESS STATE CODES

Here are the different values that the **s**, **stat** and **state** output specifiers (header "STAT" or "S") will display to describe the state of a process:

D uninterruptible sleep (usually IO)
I Idle kernel thread
R running or runnable (on run queue)
S interruptible sleep (waiting for an event to complete)
T stopped by job control signal
t stopped by debugger during the tracing
W paging (not valid since the 2.6.xx kernel)
X dead (should never be seen)
Z defunct ("zombie") process, terminated but not reaped by its parent

For BSD formats and when the **stat** keyword is used, additional characters may be displayed:

< high−priority (not nice to other users)
N low−priority (nice to other users)
L has pages locked into memory (for real−time and custom IO)
s is a session leader
l is multi-threaded (using CLONE_THREAD, like NPTL pthreads do)
+ is in the foreground process group

OBSOLETE SORT KEYS

These keys are used by the BSD **O** option (when it is used for sorting). The GNU **−−sort** option doesn't use these keys, but the specifiers described below in the **STANDARD FORMAT SPECIFIERS** section. Note that the values used in sorting are the internal values **ps** uses and not the "cooked" values used in some of the output format fields (e.g. sorting on tty will sort into device number, not according to the terminal name displayed). Pipe **ps** output into the **sort**(1) command if you want to sort the cooked values.

KEY	LONG	DESCRIPTION
c	cmd	simple name of executable
C	pcpu	cpu utilization
f	flags	flags as in long format F field
g	pgrp	process group ID
G	tpgid	controlling tty process group ID
j	cutime	cumulative user time
J	cstime	cumulative system time
k	utime	user time
m	min_flt	number of minor page faults
M	maj_flt	number of major page faults
n	cmin_flt	cumulative minor page faults
N	cmaj_flt	cumulative major page faults
o	session	session ID
p	pid	process ID
P	ppid	parent process ID
r	rss	resident set size
R	resident	resident pages
s	size	memory size in kilobytes
S	share	amount of shared pages
t	tty	the device number of the controlling tty
T	start_time	time process was started
U	uid	user ID number
u	user	user name
v	vsize	total VM size in KiB
y	priority	kernel scheduling priority

AIX FORMAT DESCRIPTORS

This **ps** supports AIX format descriptors, which work somewhat like the formatting codes of *printf*(1) and *printf*(3). For example, the normal default output can be produced with this: **ps −eo "%p %y %x %c"**. The **NORMAL** codes are described in the next section.

CODE	NORMAL	HEADER
%C	pcpu	%CPU
%G	group	GROUP
%P	ppid	PPID
%U	user	USER
%a	args	COMMAND
%c	comm	COMMAND
%g	rgroup	RGROUP
%n	nice	NI
%p	pid	PID
%r	pgid	PGID
%t	etime	ELAPSED
%u	ruser	RUSER
%x	time	TIME
%y	tty	TTY
%z	vsz	VSZ

STANDARD FORMAT SPECIFIERS

Here are the different keywords that may be used to control the output format (e.g., with option **−o**) or to sort the selected processes with the GNU−style **−−sort** option.

For example: **ps −eo pid,user,args −−sort user**

This version of **ps** tries to recognize most of the keywords used in other implementations of **ps**.

The following user−defined format specifiers may contain spaces: **args**, **cmd**, **comm**, **command**, **fname**, **ucmd**, **ucomm**, **lstart**, **bsdstart**, **start**.

Some keywords may not be available for sorting.

CODE	HEADER	DESCRIPTION
%cpu	%CPU	cpu utilization of the process in "##.#" format. Currently, it is the CPU time used divided by the time the process has been running (cputime/realtime ratio), expressed as a percentage. It will not add up to 100% unless you are lucky. (alias **pcpu**).
%mem	%MEM	ratio of the process's resident set size to the physical memory on the machine, expressed as a percentage. (alias **pmem**).
args	COMMAND	command with all its arguments as a string. Modifications to the arguments may be shown. The output in this column may contain spaces. A process marked <defunct> is partly dead, waiting to be fully destroyed by its parent. Sometimes the process args will be unavailable; when this happens, **ps** will instead print the executable name in brackets. (alias **cmd**, **command**). See also the **comm** format keyword, the **−f** option, and the **c** option. When specified last, this column will extend to the edge of the display. If **ps** can not determine display width, as when output is redirected (piped) into a file or another command, the output width is undefined (it may be 80, unlimited, determined by the **TERM** variable, and so on). The **COLUMNS** environment variable or **−−cols** option may be used to exactly determine the width in this case. The **w** or **−w** option may be also be used to adjust width.
blocked	BLOCKED	mask of the blocked signals, see *signal*(7). According to the width of the field, a 32 or 64−bit mask in hexadecimal format is displayed. (alias **sig_block**, **sigmask**).
bsdstart	START	time the command started. If the process was started less than 24 hours ago, the output format is " HH:MM", else it is " Mmm:SS" (where Mmm is the three letters of the month). See also **lstart**, **start**, **start_time**, and **stime**.
bsdtime	TIME	accumulated cpu time, user + system. The display format is usually "MMM:SS", but can be shifted to the right if the process used more than 999 minutes of cpu time.
c	C	processor utilization. Currently, this is the integer value of the percent usage over the lifetime of the process. (see **%cpu**).
caught	CAUGHT	mask of the caught signals, see *signal*(7). According to the width of the field, a 32 or 64 bits mask in hexadecimal format is displayed. (alias **sig_catch**, **sigcatch**).

cgname CGNAME display name of control groups to which the process belongs.

cgroup CGROUP display control groups to which the process belongs.

class CLS scheduling class of the process. (alias **policy**, **cls**). Field's possible values are:

–	not reported
TS	SCHED_OTHER
FF	SCHED_FIFO
RR	SCHED_RR
B	SCHED_BATCH
ISO	SCHED_ISO
IDL	SCHED_IDLE
DLN	SCHED_DEADLINE
?	unknown value

cls CLS scheduling class of the process. (alias **policy**, **cls**). Field's possible values are:

–	not reported
TS	SCHED_OTHER
FF	SCHED_FIFO
RR	SCHED_RR
B	SCHED_BATCH
ISO	SCHED_ISO
IDL	SCHED_IDLE
DLN	SCHED_DEADLINE
?	unknown value

cmd CMD see **args**. (alias **args**, **command**).

comm COMMAND command name (only the executable name). Modifications to the command name will not be shown. A process marked <defunct> is partly dead, waiting to be fully destroyed by its parent. The output in this column may contain spaces. (alias **ucmd**, **ucomm**). See also the **args** format keyword, the **–f** option, and the **c** option.
When specified last, this column will extend to the edge of the display. If **ps** can not determine display width, as when output is redirected (piped) into a file or another command, the output width is undefined (it may be 80, unlimited, determined by the **TERM** variable, and so on). The **COLUMNS** environment variable or **––cols** option may be used to exactly determine the width in this case. The **w** or **–w** option may be also be used to adjust width.

command COMMAND See **args**. (alias **args**, **command**).

cp CP per–mill (tenths of a percent) CPU usage. (see **%cpu**).

cputime TIME cumulative CPU time, "[DD–]hh:mm:ss" format. (alias **time**).

cputimes TIME cumulative CPU time in seconds (alias **times**).

drs DRS data resident set size, the amount of physical memory devoted to other than executable code.

egid	EGID	effective group ID number of the process as a decimal integer. (alias **gid**).
egroup	EGROUP	effective group ID of the process. This will be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise. (alias **group**).
eip	EIP	instruction pointer.
esp	ESP	stack pointer.
etime	ELAPSED	elapsed time since the process was started, in the form [[DD−]hh:]mm:ss.
etimes	ELAPSED	elapsed time since the process was started, in seconds.
euid	EUID	effective user ID (alias **uid**).
euser	EUSER	effective user name. This will be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise. The **n** option can be used to force the decimal representation. (alias **uname**, **user**).
exe	EXE	path to the executable. Useful if path cannot be printed via **cmd**, **comm** or **args** format options.
f	F	flags associated with the process, see the **PROCESS FLAGS** section. (alias **flag**, **flags**).
fgid	FGID	filesystem access group ID. (alias **fsgid**).
fgroup	FGROUP	filesystem access group ID. This will be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise. (alias **fsgroup**).
flag	F	see **f**. (alias **f**, **flags**).
flags	F	see **f**. (alias **f**, **flag**).
fname	COMMAND	first 8 bytes of the base name of the process's executable file. The output in this column may contain spaces.
fuid	FUID	filesystem access user ID. (alias **fsuid**).
fuser	FUSER	filesystem access user ID. This will be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise.
gid	GID	see **egid**. (alias **egid**).
group	GROUP	see **egroup**. (alias **egroup**).
ignored	IGNORED	mask of the ignored signals, see *signal*(7). According to the width of the field, a 32 or 64 bits mask in hexadecimal format is displayed. (alias **sig_ignore**, **sigignore**).

ipcns	IPCNS	Unique inode number describing the namespace the process belongs to. See *namespaces*(7).
label	LABEL	security label, most commonly used for SELinux context data. This is for the *Mandatory Access Control* ("MAC") found on high−security systems.
lstart	STARTED	time the command started. See also **bsdstart**, **start**, **start_time**, and **stime**.
lsession	SESSION	displays the login session identifier of a process, if systemd support has been included.
luid	LUID	displays Login ID associated with a process.
lwp	LWP	light weight process (thread) ID of the dispatchable entity (alias **spid**, **tid**). See **tid** for additional information.
lxc	LXC	The name of the lxc container within which a task is running. If a process is not running inside a container, a dash ('−') will be shown.
machine	MACHINE	displays the machine name for processes assigned to VM or container, if systemd support has been included.
maj_flt	MAJFLT	The number of major page faults that have occurred with this process.
min_flt	MINFLT	The number of minor page faults that have occurred with this process.
mntns	MNTNS	Unique inode number describing the namespace the process belongs to. See *namespaces*(7).
netns	NETNS	Unique inode number describing the namespace the process belongs to. See *namespaces*(7).
ni	NI	nice value. This ranges from 19 (nicest) to −20 (not nice to others), see *nice*(1). (alias **nice**).
nice	NI	see **ni**.(**alias ni**).
nlwp	NLWP	number of lwps (threads) in the process. (alias **thcount**).
numa	NUMA	The node associated with the most recently used processor. A *−1* means that NUMA information is unavailable.
nwchan	WCHAN	address of the kernel function where the process is sleeping (use **wchan** if you want the kernel function name). Running tasks will display a dash ('−') in this column.
ouid	OWNER	displays the Unix user identifier of the owner of the session of a process, if systemd support has been included.
pcpu	%CPU	see **%cpu**. (alias **%cpu**).

pending PENDING mask of the pending signals. See *signal*(7). Signals pending on the process are distinct from signals pending on individual threads. Use the **m** option or the **−m** option to see both. According to the width of the field, a 32 or 64 bits mask in hexadecimal format is displayed. (alias **sig**).

pgid PGID process group ID or, equivalently, the process ID of the process group leader. (alias **pgrp**).

pgrp PGRP see **pgid**. (alias **pgid**).

pid PID a number representing the process ID (alias **tgid**).

pidns PIDNS Unique inode number describing the namespace the process belongs to. See *namespaces*(7).

pmem %MEM see **%mem**. (alias **%mem**).

policy POL scheduling class of the process. (alias **class**, **cls**). Possible values are:

−	not reported
TS	SCHED_OTHER
FF	SCHED_FIFO
RR	SCHED_RR
B	SCHED_BATCH
ISO	SCHED_ISO
IDL	SCHED_IDLE
DLN	SCHED_DEADLINE
?	unknown value

ppid PPID parent process ID.

pri PRI priority of the process. Higher number means lower priority.

psr PSR processor that process is currently assigned to.

rgid RGID real group ID.

rgroup RGROUP real group name. This will be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise.

rss RSS resident set size, the non−swapped physical memory that a task has used (in kilobytes). (alias **rssize**, **rsz**).

rssize RSS see **rss**. (alias **rss**, **rsz**).

rsz RSZ see **rss**. (alias **rss**, **rssize**).

rtprio RTPRIO realtime priority.

ruid RUID real user ID.

ruser RUSER real user ID. This will be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise.

s	S	minimal state display (one character). See section **PROCESS STATE CODES** for the different values. See also **stat** if you want additional information displayed. (alias **state**).
sched	SCH	scheduling policy of the process. The policies SCHED_OTHER (SCHED_NORMAL), SCHED_FIFO, SCHED_RR, SCHED_BATCH, SCHED_ISO, SCHED_IDLE and SCHED_DEADLINE are respectively displayed as 0, 1, 2, 3, 4, 5 and 6.
seat	SEAT	displays the identifier associated with all hardware devices assigned to a specific workplace, if systemd support has been included.
sess	SESS	session ID or, equivalently, the process ID of the session leader. (alias **session**, **sid**).
sgi_p	P	processor that the process is currently executing on. Displays "*" if the process is not currently running or runnable.
sgid	SGID	saved group ID. (alias **svgid**).
sgroup	SGROUP	saved group name. This will be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise.
sid	SID	see **sess**. (alias **sess**, **session**).
sig	PENDING	see **pending**. (alias **pending**, **sig_pend**).
sigcatch	CAUGHT	see **caught**. (alias **caught**, **sig_catch**).
sigignore	IGNORED	see **ignored**. (alias **ignored**, **sig_ignore**).
sigmask	BLOCKED	see **blocked**. (alias **blocked**, **sig_block**).
size	SIZE	approximate amount of swap space that would be required if the process were to dirty all writable pages and then be swapped out. This number is very rough!
slice	SLICE	displays the slice unit which a process belongs to, if systemd support has been included.
spid	SPID	see **lwp**. (alias **lwp**, **tid**).
stackp	STACKP	address of the bottom (start) of stack for the process.
start	STARTED	time the command started. If the process was started less than 24 hours ago, the output format is "HH:MM:SS", else it is " Mmm dd" (where Mmm is a three−letter month name). See also **lstart**, **bsdstart**, **start_time**, and **stime**.
start_time	START	starting time or date of the process. Only the year will be displayed if the process was not started the same year **ps** was invoked, or "MmmDD" if it was not started the same day, or "HH:MM" otherwise. See also **bsdstart**, **start**, **lstart**, and **stime**.

stat	STAT	multi−character process state. See section **PROCESS STATE CODES** for the different values meaning. See also **s** and **state** if you just want the first character displayed.
state	S	see **s**. (alias **s**).
stime	STIME	see **start_time**. (alias **start_time**).
suid	SUID	saved user ID. (alias **svuid**).
supgid	SUPGID	group ids of supplementary groups, if any. See **getgroups**(2).
supgrp	SUPGRP	group names of supplementary groups, if any. See **getgroups**(2).
suser	SUSER	saved user name. This will be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise. (alias **svuser**).
svgid	SVGID	see **sgid**. (alias **sgid**).
svuid	SVUID	see **suid**. (alias **suid**).
sz	SZ	size in physical pages of the core image of the process. This includes text, data, and stack space. Device mappings are currently excluded; this is subject to change. See **vsz** and **rss**.
tgid	TGID	a number representing the thread group to which a task belongs (alias **pid**). It is the process ID of the thread group leader.
thcount	THCNT	see **nlwp**. (alias **nlwp**). number of kernel threads owned by the process.
tid	TID	the unique number representing a dispatchable entity (alias **lwp**, **spid**). This value may also appear as: a process ID (pid); a process group ID (pgrp); a session ID for the session leader (sid); a thread group ID for the thread group leader (tgid); and a tty process group ID for the process group leader (tpgid).
time	TIME	cumulative CPU time, "[DD−]HH:MM:SS" format. (alias **cputime**).
times	TIME	cumulative CPU time in seconds (alias **cputimes**).
tname	TTY	controlling tty (terminal). (alias **tt**, **tty**).
tpgid	TPGID	ID of the foreground process group on the tty (terminal) that the process is connected to, or −1 if the process is not connected to a tty.
trs	TRS	text resident set size, the amount of physical memory devoted to executable code.
tt	TT	controlling tty (terminal). (alias **tname**, **tty**).
tty	TT	controlling tty (terminal). (alias **tname**, **tt**).
ucmd	CMD	see **comm**. (alias **comm**, **ucomm**).

ucomm	COMMAND	see **comm**. (alias **comm**, **ucmd**).
uid	UID	see **euid**. (alias **euid**).
uname	USER	see **euser**. (alias **euser**, **user**).
unit	UNIT	displays unit which a process belongs to, if systemd support has been included.
user	USER	see **euser**. (alias **euser**, **uname**).
userns	USERNS	Unique inode number describing the namespace the process belongs to. See *namespaces*(7).
utsns	UTSNS	Unique inode number describing the namespace the process belongs to. See *namespaces*(7).
uunit	UUNIT	displays user unit which a process belongs to, if systemd support has been included.
vsize	VSZ	see **vsz**. (alias **vsz**).
vsz	VSZ	virtual memory size of the process in KiB (1024−byte units). Device mappings are currently excluded; this is subject to change. (alias **vsize**).
wchan	WCHAN	name of the kernel function in which the process is sleeping, a "−" if the process is running, or a "*" if the process is multi−threaded and **ps** is not displaying threads.

ENVIRONMENT VARIABLES

The following environment variables could affect **ps**:

COLUMNS
Override default display width.

LINES
Override default display height.

PS_PERSONALITY
Set to one of posix, old, linux, bsd, sun, digital... (see section **PERSONALITY** below).

CMD_ENV
Set to one of posix, old, linux, bsd, sun, digital... (see section **PERSONALITY** below).

I_WANT_A_BROKEN_PS
Force obsolete command line interpretation.

LC_TIME
Date format.

PS_COLORS
Not currently supported.

PS_FORMAT
Default output format override. You may set this to a format string of the type used for the **−o** option. The **DefSysV** and **DefBSD** values are particularly useful.

POSIXLY_CORRECT
Don't find excuses to ignore bad "features".

POSIX2
When set to "on", acts as **POSIXLY_CORRECT**.

UNIX95
Don't find excuses to ignore bad "features".

_XPG
Cancel **CMD_ENV**=*irix* non−standard behavior.

In general, it is a bad idea to set these variables. The one exception is **CMD_ENV** or **PS_PERSONALITY**, which could be set to Linux for normal systems. Without that setting, **ps** follows the useless and bad parts of the Unix98 standard.

PERSONALITY

390	like the OS/390 OpenEdition **ps**
aix	like AIX **ps**
bsd	like FreeBSD **ps** (totally non−standard)
compaq	like Digital Unix **ps**
debian	like the old Debian **ps**
digital	like Tru64 (was Digital Unix, was OSF/1) **ps**
gnu	like the old Debian **ps**
hp	like HP−UX **ps**
hpux	like HP−UX **ps**
irix	like Irix **ps**
linux	***** **recommended** *****
old	like the original Linux **ps** (totally non−standard)
os390	like OS/390 Open Edition **ps**
posix	standard
s390	like OS/390 Open Edition **ps**
sco	like SCO **ps**
sgi	like Irix **ps**
solaris2	like Solaris 2+ (SunOS 5) **ps**
sunos4	like SunOS 4 (Solaris 1) **ps** (totally non−standard)
svr4	standard
sysv	standard
tru64	like Tru64 (was Digital Unix, was OSF/1) **ps**
unix	standard
unix95	standard
unix98	standard

SEE ALSO

pgrep(1), **pstree**(1), **top**(1), **proc**(5).

STANDARDS

This **ps** conforms to:

1 Version 2 of the Single Unix Specification
2 The Open Group Technical Standard Base Specifications, Issue 6
3 IEEE Std 1003.1, 2004 Edition
4 X/Open System Interfaces Extension [UP XSI]
5 ISO/IEC 9945:2003

AUTHOR

ps was originally written by Branko Lankester ⟨lankeste@fwi.uva.nl⟩. Michael K. Johnson ⟨johnsonm@redhat.com⟩ re−wrote it significantly to use the proc filesystem, changing a few things in the process. Michael Shields ⟨mjshield@nyx.cs.du.edu⟩ added the pid−list feature. Charles Blake ⟨cblake@bbn.com⟩ added multi−level sorting, the dirent−style library, the device name−to−number mmaped database, the

approximate binary search directly on System.map, and many code and documentation cleanups. David Mossberger−Tang wrote the generic BFD support for psupdate. Albert Cahalan ⟨albert@users.sf.net⟩ rewrote ps for full Unix98 and BSD support, along with some ugly hacks for obsolete and foreign syntax.

Please send bug reports to ⟨procps@freelists.org⟩. No subscription is required or suggested.

NAME

time − overview of time and timers

DESCRIPTION

Real time and process time

Real time is defined as time measured from some fixed point, either from a standard point in the past (see the description of the Epoch and calendar time below), or from some point (e.g., the start) in the life of a process (*elapsed time*).

Process time is defined as the amount of CPU time used by a process. This is sometimes divided into *user* and *system* components. User CPU time is the time spent executing code in user mode. System CPU time is the time spent by the kernel executing in system mode on behalf of the process (e.g., executing system calls). The **time**(1) command can be used to determine the amount of CPU time consumed during the execution of a program. A program can determine the amount of CPU time it has consumed using **times**(2), **getrusage**(2), or **clock**(3).

The hardware clock

Most computers have a (battery-powered) hardware clock which the kernel reads at boot time in order to initialize the software clock. For further details, see **rtc**(4) and **hwclock**(8).

The software clock, HZ, and jiffies

The accuracy of various system calls that set timeouts, (e.g., **select**(2), **sigtimedwait**(2)) and measure CPU time (e.g., **getrusage**(2)) is limited by the resolution of the *software clock*, a clock maintained by the kernel which measures time in *jiffies*. The size of a jiffy is determined by the value of the kernel constant *HZ*.

The value of *HZ* varies across kernel versions and hardware platforms. On i386 the situation is as follows: on kernels up to and including 2.4.x, HZ was 100, giving a jiffy value of 0.01 seconds; starting with 2.6.0, HZ was raised to 1000, giving a jiffy of 0.001 seconds. Since kernel 2.6.13, the HZ value is a kernel configuration parameter and can be 100, 250 (the default) or 1000, yielding a jiffies value of, respectively, 0.01, 0.004, or 0.001 seconds. Since kernel 2.6.20, a further frequency is available: 300, a number that divides evenly for the common video frame rates (PAL, 25 HZ; NTSC, 30 HZ).

The **times**(2) system call is a special case. It reports times with a granularity defined by the kernel constant *USER_HZ*. User-space applications can determine the value of this constant using *sysconf(_SC_CLK_TCK)*.

System and process clocks; time namespaces

The kernel supports a range of clocks that measure various kinds of elapsed and virtual (i.e., consumed CPU) time. These clocks are described in **clock_gettime**(2). A few of the clocks are settable using **clock_settime**(2). The values of certain clocks are virtualized by time namespaces; see **time_namespaces**(7).

High-resolution timers

Before Linux 2.6.21, the accuracy of timer and sleep system calls (see below) was also limited by the size of the jiffy.

Since Linux 2.6.21, Linux supports high-resolution timers (HRTs), optionally configurable via **CONFIG_HIGH_RES_TIMERS**. On a system that supports HRTs, the accuracy of sleep and timer system calls is no longer constrained by the jiffy, but instead can be as accurate as the hardware allows (microsecond accuracy is typical of modern hardware). You can determine whether high-resolution timers are supported by checking the resolution returned by a call to **clock_getres**(2) or looking at the "resolution" entries in */proc/timer_list*.

HRTs are not supported on all hardware architectures. (Support is provided on x86, arm, and powerpc, among others.)

The Epoch

UNIX systems represent time in seconds since the *Epoch*, 1970-01-01 00:00:00 +0000 (UTC).

A program can determine the *calendar time* via the **clock_gettime**(2) **CLOCK_REALTIME** clock, which returns time (in seconds and nanoseconds) that have elapsed since the Epoch; **time**(2) provides similar information, but only with accuracy to the nearest second. The system time can be changed using

clock_settime(2).

Broken-down time

Certain library functions use a structure of type *tm* to represent *broken-down time*, which stores time value separated out into distinct components (year, month, day, hour, minute, second, etc.). This structure is described in **ctime**(3), which also describes functions that convert between calendar time and broken-down time. Functions for converting between broken-down time and printable string representations of the time are described in **ctime**(3), **strftime**(3), and **strptime**(3).

Sleeping and setting timers

Various system calls and functions allow a program to sleep (suspend execution) for a specified period of time; see **nanosleep**(2), **clock_nanosleep**(2), and **sleep**(3).

Various system calls allow a process to set a timer that expires at some point in the future, and optionally at repeated intervals; see **alarm**(2), **getitimer**(2), **timerfd_create**(2), and **timer_create**(2).

Timer slack

Since Linux 2.6.28, it is possible to control the "timer slack" value for a thread. The timer slack is the length of time by which the kernel may delay the wake-up of certain system calls that block with a timeout. Permitting this delay allows the kernel to coalesce wake-up events, thus possibly reducing the number of system wake-ups and saving power. For more details, see the description of **PR_SET_TIMERSLACK** in **prctl**(2).

SEE ALSO

date(1), **time**(1), **timeout**(1), **adjtimex**(2), **alarm**(2), **clock_gettime**(2), **clock_nanosleep**(2), **getitimer**(2), **getrlimit**(2), **getrusage**(2), **gettimeofday**(2), **nanosleep**(2), **stat**(2), **time**(2), **timer_create**(2), **timerfd_create**(2), **times**(2), **utime**(2), **adjtime**(3), **clock**(3), **clock_getcpuclockid**(3), **ctime**(3), **ntp_adjtime**(3), **ntp_gettime**(3), **pthread_getcpuclockid**(3), **sleep**(3), **strftime**(3), **strptime**(3), **timeradd**(3), **usleep**(3), **rtc**(4), **time_namespaces**(7), **hwclock**(8)

COLOPHON

This page is part of release 5.10 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man−pages/.