NAME

ps - process status

SYNOPSIS

ps [**--libxo**] [**-aCcdefHhjlmrSTuvwXxZ**] [**-O** *fmt* | **-o** *fmt*] [**-D** *up* | *down* | *both*] [**-G** *gid*[,*gid*...]]
[**-J** *jid*[,*jid*...]] [**-M** *core*] [**-N** *system*] [**-p** *pid*[,*pid*...]] [**-t** *tty*[,*tty*...]] [**-U** *user*[,*user*...]]
ps [**--libxo**] **-L**

DESCRIPTION

The **ps** utility displays a header line, followed by lines containing information about all of your processes that have controlling terminals. If the **-x** options is specified, **ps** will also display processes that do not have controlling terminals.

A different set of processes can be selected for display by using any combination of the **-a**, **-G**, **-J**, **-p**, **-T**, **-t**, and **-U** options. If more than one of these options are given, then **ps** will select all processes which are matched by at least one of the given options.

For the processes which have been selected for display, **ps** will usually display one line per process. The **-H** option may result in multiple output lines (one line per thread) for some processes. By default all of these output lines are sorted first by controlling terminal, then by process ID. The **-m**, **-r**, **-u**, and **-v** options will change the sort order. If more than one sorting option was given, then the selected processes will be sorted by the last sorting option which was specified.

For the processes which have been selected for display, the information to display is selected based on a set of keywords (see the **-L**, **-O**, and **-o** options). The default output format includes, for each process, the process' ID, controlling terminal, state, CPU time (including both user and system time) and associated command.

If the **ps** process is associated with a terminal, the default output width is that of the terminal; otherwise the output width is unlimited. See also the **-w** option.

The options are as follows:

--libxo
: Generate output via libxo(3) in a selection of different human and machine readable formats. See xo_parse_args(3) for details on command line arguments.

-a
: Display information about other users' processes as well as your own. If the *security.bsd.see_other_uids* sysctl is set to zero, this option is honored only if the UID of the user is 0.

-c Change the "command" column output to just contain the executable name, rather than the full command line.

-C Change the way the CPU percentage is calculated by using a "raw" CPU calculation that ignores "resident" time (this normally has no effect).

-d Arrange processes into descendancy order and prefix each command with indentation text showing sibling and parent/child relationships as a tree. If either of the **-m** and **-r** options are also used, they control how sibling processes are sorted relative to each other. Note that this option has no effect if the "command" column is not the last column displayed.

-D Expand the list of selected processes based on the process tree. "UP" will add the ancestor processes, "DOWN" will add the descendant processes, and "BOTH" will add both the ancestor and the descendant processes. **-D** does not imply **-d**, but works well with it.

-e Display the environment as well.

-f Show command-line and environment information about swapped out processes. This option is honored only if the UID of the user is 0.

-G Display information about processes which are running with the specified real group IDs.

-H Show all of the threads associated with each process.

-h Repeat the information header as often as necessary to guarantee one header per page of information.

-j Print information associated with the following keywords: **user**, **pid**, **ppid**, **pgid**, **sid**, **jobc**, **state**, **tt**, **time**, and **command**.

-J Display information about processes which match the specified jail IDs. This may be either the **jid** or **name** of the jail. Use **-J 0** to display only host processes. This flag implies **-x** by default.

-L List the set of keywords available for the **-O** and **-o** options.

-l Display information associated with the following keywords: **uid**, **pid**, **ppid**, **cpu**, **pri**, **nice**, **vsz**, **rss**, **mwchan**, **state**, **tt**, **time**, and **command**.

-M Extract values associated with the name list from the specified core instead of the currently running system.

-m Sort by memory usage, instead of the combination of controlling terminal and process ID.

-N Extract the name list from the specified system instead of the default, which is the kernel image the system has booted from.

-O Add the information associated with the space or comma separated list of keywords specified, after the process ID, in the default information display. Keywords may be appended with an equals (‘=’) sign and a string. This causes the printed header to use the specified string instead of the standard header.

-o Display information associated with the space or comma separated list of keywords specified. The last keyword in the list may be appended with an equals (‘=’) sign and a string that spans the rest of the argument, and can contain space and comma characters. This causes the printed header to use the specified string instead of the standard header. Multiple keywords may also be given in the form of more than one **-o** option. So the header texts for multiple keywords can be changed. If all keywords have empty header texts, no header line is written.

-p Display information about processes which match the specified process IDs.

-r Sort by current CPU usage, instead of the combination of controlling terminal and process ID.

-S Change the way the process times, namely cputime, systime, and usertime, are calculated by summing all exited children to their parent process.

-T Display information about processes attached to the device associated with the standard input.

-t Display information about processes attached to the specified terminal devices. Full pathnames, as well as abbreviations (see explanation of the **tt** keyword) can be specified.

-U Display the processes belonging to the specified usernames.

-u Display information associated with the following keywords: **user**, **pid**, **%cpu**, **%mem**, **vsz**, **rss**, **tt**, **state**, **start**, **time**, and **command**. The **-u** option implies the **-r** option.

-v Display information associated with the following keywords: **pid**, **state**, **time**, **sl**, **re**, **pagein**, **vsz**, **rss**, **lim**, **tsiz**, **%cpu**, **%mem**, and **command**. The **-v** option implies the **-m** option.

-w Use at least 132 columns to display information, instead of the default which is the window size if **ps** is associated with a terminal. If the **-w** option is specified more than once, **ps** will use as many columns as necessary without regard for the window size. Note that this option has no

effect if the "command" column is not the last column displayed.

-X When displaying processes matched by other options, skip any processes which do not have a controlling terminal. This is the default behaviour.

-x When displaying processes matched by other options, include processes which do not have a controlling terminal. This is the opposite of the **-X** option. If both **-X** and **-x** are specified in the same command, then **ps** will use the one which was specified last.

-Z Add mac(4) label to the list of keywords for which **ps** will display information.

A complete list of the available keywords are listed below. Some of these keywords are further specified as follows:

%cpu The CPU utilization of the process; this is a decaying average over up to a minute of previous (real) time. Since the time base over which this is computed varies (since processes may be very young) it is possible for the sum of all **%cpu** fields to exceed 100%.

%mem The percentage of real memory used by this process.

class Login class associated with the process.

flags The flags associated with the process as in the include file <*sys/proc.h*>:

P_ADVLOCK	0x00001	Process may hold a POSIX advisory lock
P_CONTROLT	0x00002	Has a controlling terminal
P_KPROC	0x00004	Kernel process
P_PPWAIT	0x00010	Parent is waiting for child to exec/exit
P_PROFIL	0x00020	Has started profiling
P_STOPPROF	0x00040	Has thread in requesting to stop prof
P_HADTHREADS	0x00080	Has had threads (no cleanup shortcuts)
P_SUGID	0x00100	Had set id privileges since last exec
P_SYSTEM	0x00200	System proc: no sigs, stats or swapping
P_SINGLE_EXIT	0x00400	Threads suspending should exit, not wait
P_TRACED	0x00800	Debugged process being traced
P_WAITED	0x01000	Someone is waiting for us
P_WEXIT	0x02000	Working on exiting
P_EXEC	0x04000	Process called exec
P_WKILLED	0x08000	Killed, shall go to kernel/user boundary ASAP
P_CONTINUED	0x10000	Proc has continued from a stopped state

P_STOPPED_SIG	0x20000	Stopped due to SIGSTOP/SIGTSTP
P_STOPPED_TRACE	0x40000	Stopped because of tracing
P_STOPPED_SINGLE	0x80000	Only one thread can continue
P_PROTECTED	0x100000	Do not kill on memory overcommit
P_SIGEVENT	0x200000	Process pending signals changed
P_SINGLE_BOUNDARY	0x400000	Threads should suspend at user boundary
P_HWPMC	0x800000	Process is using HWPMCs
P_JAILED	0x1000000	Process is in jail
P_TOTAL_STOP	0x2000000	Stopped for system suspend
P_INEXEC	0x4000000	Process is in execve(2)
P_STATCHILD	0x8000000	Child process stopped or exited
P_INMEM	0x10000000	Loaded into memory
P_SWAPPINGOUT	0x20000000	Process is being swapped out
P_SWAPPINGIN	0x40000000	Process is being swapped in
P_PPTRACE	0x80000000	Vforked child issued ptrace(PT_TRACEME)

flags2 The flags kept in *p_flag2* associated with the process as in the include file <*sys/proc.h*>:

P2_INHERIT_PROTECTED	0x00000001	New children get P_PROTECTED
P2_NOTRACE	0x00000002	No ptrace(2) attach or coredumps
P2_NOTRACE_EXEC	0x00000004	Keep P2_NOPTRACE on execve(2)
P2_AST_SU	0x00000008	Handles SU ast for kthreads
P2_PTRACE_FSTP	0x00000010	SIGSTOP from PT_ATTACH not yet handled

label The MAC label of the process.

lim The soft limit on memory used, specified via a call to setrlimit(2).

lstart The exact time the command started, using the ‘%c’ format described in strftime(3).

lockname The name of the lock that the process is currently blocked on. If the name is invalid or unknown, then "???" is displayed.

logname The login name associated with the session the process is in (see getlogin(2)).

mwchan The event name if the process is blocked normally, or the lock name if the process is blocked on a lock. See the wchan and lockname keywords for details.

nice The process scheduling increment (see setpriority(2)).

rss the real memory (resident set) size of the process (in 1024 byte units).

start The time the command started. If the command started less than 24 hours ago, the start time is displayed using the "%H:%M" format described in strftime(3). If the command started less than 7 days ago, the start time is displayed using the "%a%H" format. Otherwise, the start time is displayed using the "%e%b%y" format.

state The state is given by a sequence of characters, for example, "RWNA". The first character indicates the run state of the process:

D	Marks a process in disk (or other short term, uninterruptible) wait.
I	Marks a process that is idle (sleeping for longer than about 20 seconds).
L	Marks a process that is waiting to acquire a lock.
R	Marks a runnable process.
S	Marks a process that is sleeping for less than about 20 seconds.
T	Marks a stopped process.
W	Marks an idle interrupt thread.
Z	Marks a dead process (a "zombie").

Additional characters after these, if any, indicate additional state information:

+	The process is in the foreground process group of its control terminal.
<	The process has raised CPU scheduling priority.
C	The process is in capsicum(4) capability mode.
E	The process is trying to exit.
J	Marks a process which is in jail(2). The hostname of the prison can be found in */proc/<pid>/status*.
L	The process has pages locked in core (for example, for raw I/O).
N	The process has reduced CPU scheduling priority (see setpriority(2)).
s	The process is a session leader.
V	The process' parent is suspended during a vfork(2), waiting for the process to exec or exit.
W	The process is swapped out.
X	The process is being traced or debugged.

tt An abbreviation for the pathname of the controlling terminal, if any. The abbreviation consists of the three letters following */dev/tty*, or, for pseudo-terminals, the corresponding entry in */dev/pts*. This is followed by a '-' if the process can no longer reach that controlling terminal (i.e., it has been revoked). A '-' without a preceding two letter abbreviation or pseudo-terminal device number indicates a process which never had a controlling terminal.

The full pathname of the controlling terminal is available via the **tty** keyword.

wchan The event (an address in the system) on which a process waits. When printed numerically, the initial part of the address is trimmed off and the result is printed in hex, for example, 0x80324000 prints as 324000.

When printing using the command keyword, a process that has exited and has a parent that has not yet waited for the process (in other words, a zombie) is listed as "<defunct>", and a process which is blocked while trying to exit is listed as "<exiting>". If the arguments cannot be located (usually because it has not been set, as is the case of system processes and/or kernel threads) the command name is printed within square brackets. The **ps** utility first tries to obtain the arguments cached by the kernel (if they were shorter than the value of the *kern.ps_arg_cache_limit* sysctl). The process can change the arguments shown with setproctitle(3). Otherwise, **ps** makes an educated guess as to the file name and arguments given when the process was created by examining memory or the swap area. The method is inherently somewhat unreliable and in any event a process is entitled to destroy this information. The ucomm (accounting) keyword can, however, be depended on. If the arguments are unavailable or do not agree with the ucomm keyword, the value for the ucomm keyword is appended to the arguments in parentheses.

KEYWORDS

The following is a complete list of the available keywords and their meanings. Several of them have aliases (keywords which are synonyms).

%cpu percentage CPU usage (alias **pcpu**)
%mem percentage memory usage (alias **pmem**)
acflag accounting flag (alias **acflg**)
args command and arguments
class login class
comm command
command
command and arguments
cow number of copy-on-write faults
cpu The processor number on which the process is executing (visible only on SMP systems).
dsiz data size (in Kbytes)
emul system-call emulation environment (ABI)
etime elapsed running time, format "[days-][hours:]minutes:seconds"
etimes elapsed running time, in decimal integer seconds
fib default FIB number, see setfib(1)
flags the process flags, in hexadecimal (alias **f**)
flags2 the additional set of process flags, in hexadecimal (alias **f2**)

gid effective group ID (alias **egid**)
group group name (from egid) (alias **egroup**)
inblk total blocks read (alias **inblock**)
jail jail name
jid jail ID
jobc job control count
ktrace tracing flags
label MAC label
lim memoryuse limit
lockname
lock currently blocked on (as a symbolic name)
logname login name of user who started the session
lstart time started
lwp thread (light-weight process) ID (alias **tid**)
majflt total page faults
minflt total page reclaims
msgrcv total messages received (reads from pipes/sockets)
msgsnd total messages sent (writes on pipes/sockets)
mwchan wait channel or lock currently blocked on
nice nice value (alias **ni**)
nivcsw total involuntary context switches
nlwp number of threads (light-weight processes) tied to a process
nsigs total signals taken (alias **nsignals**)
nswap total swaps in/out
nvcsw total voluntary context switches
nwchan wait channel (as an address)
oublk total blocks written (alias **oublock**)
paddr process pointer
pagein pageins (same as majflt)
pgid process group number
pid process ID
ppid parent process ID
pri scheduling priority
re core residency time (in seconds; 127 = infinity)
rgid real group ID
rgroup group name (from rgid)
rss resident set size
rtprio realtime priority (see rtprio(1))
ruid real user ID
ruser user name (from ruid)

sid session ID
sig pending signals (alias **pending**)
sigcatch caught signals (alias **caught**)
sigignore ignored signals (alias **ignored**)
sigmask blocked signals (alias **blocked**)
sl sleep time (in seconds; 127 = infinity)
ssiz stack size (in Kbytes)
start time started
state symbolic process state (alias **stat**)
svgid saved gid from a setgid executable
svuid saved UID from a setuid executable
systime accumulated system CPU time
tdaddr thread address
tdname thread name
tdev control terminal device number
time accumulated CPU time, user + system (alias **cputime**)
tpgid control terminal process group ID
tracer tracer process ID
tsid control terminal session ID
tsiz text size (in Kbytes)
tt control terminal name (two letter abbreviation)
tty full name of control terminal
ucomm name to be used for accounting
uid effective user ID (alias **euid**)
upr scheduling priority on return from system call (alias **usrpri**)
uprocp process pointer
user user name (from UID)
usertime accumulated user CPU time
vmaddr vmspace pointer
vsz virtual size in Kbytes (alias **vsize**)
wchan wait channel (as a symbolic name)
xstat exit or stop status (valid only for stopped or zombie process)

Note that the **pending** column displays bitmask of signals pending in the process queue when **-H** option is not specified, otherwise the per-thread queue of pending signals is shown.

ENVIRONMENT

The following environment variables affect the execution of **ps**:

COLUMNS If set, specifies the user's preferred output width in column positions. By default, **ps**

attempts to automatically determine the terminal width.

FILES

/boot/kernel/kernel default system namelist

EXIT STATUS

The **ps** utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

Display information on all system processes:

```
$ ps -auxw
```

SEE ALSO

kill(1), pgrep(1), pkill(1), procstat(1), w(1), kvm(3), libxo(3), strftime(3), xo_parse_args(3), mac(4), procfs(5), pstat(8), sysctl(8), mutex(9)

STANDARDS

For historical reasons, the **ps** utility under FreeBSD supports a different set of options from what is described by IEEE Std 1003.2 ("POSIX.2"), and what is supported on non-BSD operating systems.

HISTORY

The **ps** command appeared in Version 3 AT&T UNIX in section 8 of the manual.

BUGS

Since **ps** cannot run faster than the system and is run as any other scheduled process, the information it displays can never be exact.

The **ps** utility does not correctly display argument lists containing multibyte characters.