Package 'ps'

July 22, 2018

Version 1.0.0

Title List, Query, Manipulate System Processes

Description List, query and manipulate all system processes, on 'Windows',
'Linux' and 'macOS'.

License BSD_3_clause + file LICENSE

URL https://github.com/r-lib/ps#readme

BugReports https://github.com/r-lib/ps/issues

Encoding UTF-8

Depends R (>= 3.1)

Imports utils

Suggests callr,
covr,
processx,
testthat,
tibble

RoxygenNote 6.0.1

Roxygen list(markdown = TRUE)

Biarch true

R topics documented:

ps	*Process table*

Description

Process table

Usage

```
ps(user = NULL, after = NULL)
```

Arguments

user	Username, to filter the results to matching processes.
after	Start time (POSIXt), to filter the results to processes that started after this.

Value

Data frame (tibble), see columns below.

Columns:

- `pid`: Process ID.
- `ppid`: Process ID of parent process.
- `name`: Process name.
- `username`: Name of the user (real uid on POSIX).
- `status`: I.e. *running*, *sleeping*, etc.
- `user`: User CPU time.
- `system`: System CPU time.
- `rss`: Resident set size, the amount of memory the process currently uses. Does not include memory that is swapped out. It does include shared libraries.
- `vms`: Virtual memory size. All memory the process has access to.
- `created`: Time stamp when the process was created.
- `ps_handle`: ps_handle objects, in a list column.

ps_boot_time	*Boot time of the system*

Description

Boot time of the system

Usage

```
ps_boot_time()
```

Value

A `POSIXct` object.

ps_children	*List of child processes (process objects) of the process. Note that this typically requires enumerating all processes on the system, so it is a costly operation.*

Description

List of child processes (process objects) of the process. Note that this typically requires enumerating all processes on the system, so it is a costly operation.

Usage

```
ps_children(p, recursive = FALSE)
```

Arguments

p	Process handle.
recursive	Whether to include the children of the children, etc.

Value

List of `ps_handle` objects.

Examples

```
p <- ps_parent(ps_handle())
ps_children(p)
```

See Also

Other process handle functions: `ps_cmdline`, `ps_cpu_times`, `ps_create_time`, `ps_cwd`, `ps_environ`, `ps_exe`, `ps_handle`, `ps_is_running`, `ps_kill`, `ps_memory_info`, `ps_name`, `ps_num_threads`, `ps_pid`, `ps_ppid`, `ps_resume`, `ps_send_signal`, `ps_status`, `ps_suspend`, `ps_terminal`, `ps_terminate`, `ps_uids`, `ps_username`

---

ps_cmdline *Command line of the process*

Description

Command line of the process, i.e. the executable and the command line arguments, in a character vector. On Unix the program might change its command line, and some programs actually do it.

Usage

```
ps_cmdline(p)
```

Arguments

p   Process handle.

Details

For a zombie process it throws a `zombie_process` error.

Value

Character vector.

Examples

```
p <- ps_handle()
p
ps_name(p)
ps_exe(p)
ps_cmdline(p)
```

See Also

Other process handle functions: ps_children, ps_cpu_times, ps_create_time, ps_cwd, ps_environ, ps_exe, ps_handle, ps_is_running, ps_kill, ps_memory_info, ps_name, ps_num_threads, ps_pid, ps_ppid, ps_resume, ps_send_signal, ps_status, ps_suspend, ps_terminal, ps_terminate, ps_uids, ps_username

---

ps_cpu_times *CPU times of the process*

Description

All times are measued in seconds:

- user: Amount of time that this process has been scheduled in user mode.
- system: Amount of time that this process has been scheduled in kernel mode
- childen_user: On Linux, amount of time that this process's waited-for children have been scheduled in user mode.
- children_system: On Linux, Amount of time that this process's waited-for children have been scheduled in kernel mode.

Usage

```
ps_cpu_times(p)
```

Arguments

p   Process handle.

Details

Throws a `zombie_process()` error for zombie processes.

Value

Named real vector or length four: `user`, `system`, `childen_user`, `children_system`. The last two are `NA` on non-Linux systems.

Examples

```
p <- ps_handle()
p
ps_cpu_times(p)
proc.time()
```

See Also

Other process handle functions: `ps_children`, `ps_cmdline`, `ps_create_time`, `ps_cwd`, `ps_environ`, `ps_exe`, `ps_handle`, `ps_is_running`, `ps_kill`, `ps_memory_info`, `ps_name`, `ps_num_threads`, `ps_pid`, `ps_ppid`, `ps_resume`, `ps_send_signal`, `ps_status`, `ps_suspend`, `ps_terminal`, `ps_terminate`, `ps_uids`, `ps_username`

---

ps_create_time   *Start time of a process*

Description

The pid and the start time pair serves as the identifier of the process, as process ids might be reused, but the chance of starting two processes with identical ids within the resolution of the timer is minimal.

Usage

```
ps_create_time(p)
```

Arguments

p   Process handle.

Details

This function works even if the process has already finished.

Value

POSIXct object, start time, in GMT.

Examples

```
p <- ps_handle()
p
ps_create_time(p)
```

See Also

Other process handle functions: `ps_children`, `ps_cmdline`, `ps_cpu_times`, `ps_cwd`, `ps_environ`, `ps_exe`, `ps_handle`, `ps_is_running`, `ps_kill`, `ps_memory_info`, `ps_name`, `ps_num_threads`, `ps_pid`, `ps_ppid`, `ps_resume`, `ps_send_signal`, `ps_status`, `ps_suspend`, `ps_terminal`, `ps_terminate`, `ps_uids`, `ps_username`

---

ps_cwd   *Process current working directory as an absolute path.*

Description

For a zombie process it throws a `zombie_process` error.

Usage

```
ps_cwd(p)
```

Arguments

p   Process handle.

Value

String scalar.

Examples

```
p <- ps_handle()
p
ps_cwd(p)
```

See Also

Other process handle functions: `ps_children`, `ps_cmdline`, `ps_cpu_times`, `ps_create_time`, `ps_environ`, `ps_exe`, `ps_handle`, `ps_is_running`, `ps_kill`, `ps_memory_info`, `ps_name`, `ps_num_threads`, `ps_pid`, `ps_ppid`, `ps_resume`, `ps_send_signal`, `ps_status`, `ps_suspend`, `ps_terminal`, `ps_terminate`, `ps_uids`, `ps_username`

---

ps_environ   *Environment variables of a process*

Description

ps_environ() returns the environment variables of the process, in a named vector, similarly to the return value of Sys.getenv() (without arguments).

Usage

```
ps_environ(p)

ps_environ_raw(p)
```

Arguments

p   Process handle.

Details

Note: this usually does not reflect changes made after the process started.

ps_environ_raw() is similar to p$environ() but returns the unparsed "var=value" strings. This is faster, and sometimes good enough.

These functions throw a zombie_process error for zombie processes.

Value

ps_environ() returns a named character vector (that has a Dlist class, so it is printed nicely), ps_environ_raw() returns a character vector.

Examples

```
p <- ps_handle()
p
env <- ps_environ(p)
env[["R_HOME"]]
```

See Also

Other process handle functions: ps_children, ps_cmdline, ps_cpu_times, ps_create_time, ps_cwd, ps_exe, ps_handle, ps_is_running, ps_kill, ps_memory_info, ps_name, ps_num_threads, ps_pid, ps_ppid, ps_resume, ps_send_signal, ps_status, ps_suspend, ps_terminal, ps_terminate, ps_uids, ps_username

Other process handle functions: ps_children, ps_cmdline, ps_cpu_times, ps_create_time, ps_cwd, ps_exe, ps_handle, ps_is_running, ps_kill, ps_memory_info, ps_name, ps_num_threads, ps_pid, ps_ppid, ps_resume, ps_send_signal, ps_status, ps_suspend, ps_terminal, ps_terminate, ps_uids, ps_username

---

ps_exe   *Full path of the executable of a process*

Description

Path to the executable of the process. May also be an empty string or NA if it cannot be determined.

Usage

```
ps_exe(p)
```

Arguments

p   Process handle.

Details

For a zombie process it throws a zombie_process error.

Value

Character scalar.

Examples

```
p <- ps_handle()
p
ps_name(p)
ps_exe(p)
ps_cmdline(p)
```

See Also

Other process handle functions: ps_children, ps_cmdline, ps_cpu_times, ps_create_time, ps_cwd, ps_environ, ps_handle, ps_is_running, ps_kill, ps_memory_info, ps_name, ps_num_threads, ps_pid, ps_ppid, ps_resume, ps_send_signal, ps_status, ps_suspend, ps_terminal, ps_terminate, ps_uids, ps_username

---

ps_handle   *Create a process handle*

Description

Create a process handle

Usage

```
ps_handle(pid = NULL, time = NULL)

## S3 method for class 'ps_handle'
format(x, ...)

## S3 method for class 'ps_handle'
print(x, ...)
```

Arguments

pid	Process id. Integer scalar. NULL means the current R process.
time	Start time of the process. Usually NULL and ps will query the start time.
x	Process handle.
...	Not used currently.

Value

ps_handle() returns a process handle (class ps_handle).

Examples

```
p <- ps_handle()
p
```

See Also

Other process handle functions: ps_children, ps_cmdline, ps_cpu_times, ps_create_time, ps_cwd, ps_environ, ps_exe, ps_is_running, ps_kill, ps_memory_info, ps_name, ps_num_threads, ps_pid, ps_ppid, ps_resume, ps_send_signal, ps_status, ps_suspend, ps_terminal, ps_terminate, ps_uids, ps_username

---

ps_is_running   *Checks whether a process is running*

Description

It returns FALSE if the process has already finished.

Usage

```
ps_is_running(p)
```

Arguments

p	Process handle.

Details

It uses the start time of the process to work around pid reuse. I.e.

Value

Logical scalar.

Examples

```
p <- ps_handle()
p
ps_is_running(p)
```

See Also

Other process handle functions: ps_children, ps_cmdline, ps_cpu_times, ps_create_time, ps_cwd, ps_environ, ps_exe, ps_handle, ps_kill, ps_memory_info, ps_name, ps_num_threads, ps_pid, ps_ppid, ps_resume, ps_send_signal, ps_status, ps_suspend, ps_terminal, ps_terminate, ps_uids, ps_username

---

ps_kill   *Kill a process*

Description

Kill the current process with SIGKILL pre-emptively checking whether PID has been reused. On Windows it uses TerminateProcess().

Usage

```
ps_kill(p)
```

Arguments

p   Process handle.

Examples

```
px <- processx::process$new("sleep", "10")
p <- ps_handle(px$get_pid())
p
ps_kill(p)
p
ps_is_running(p)
px$get_exit_status()
```

See Also

Other process handle functions: ps_children, ps_cmdline, ps_cpu_times, ps_create_time, ps_cwd, ps_environ, ps_exe, ps_handle, ps_is_running, ps_memory_info, ps_name, ps_num_threads, ps_pid, ps_ppid, ps_resume, ps_send_signal, ps_status, ps_suspend, ps_terminal, ps_terminate, ps_uids, ps_username

---

ps_memory_info   *Memory usage information*

Description

A list with information about memory usage. Portable fields:

- rss: "Resident Set Size", this is the non-swapped physical memory a process has used. On UNIX it matches "top"'s RES column (see doc). On Windows this is an alias for wset field and it matches "Memory" column of taskmgr.exe.

- vmem: "Virtual Memory Size", this is the total amount of virtual memory used by the process. On UNIX it matches "top"'s VIRT column (see doc). On Windows this is an alias for the pagefile field and it matches the "Working set (memory)" column of taskmgr.exe.

Usage

```
ps_memory_info(p)
```

Arguments

p   Process handle.

Details

Non-portable fields:

- shared: (Linux) memory that could be potentially shared with other processes. This matches "top"'s SHR column (see doc).

- text: (Linux): aka TRS (text resident set) the amount of memory devoted to executable code. This matches "top"'s CODE column (see doc).

- data: (Linux): aka DRS (data resident set) the amount of physical memory devoted to other than executable code. It matches "top"'s DATA column (see doc).

- lib: (Linux): the memory used by shared libraries.

- dirty: (Linux): the number of dirty pages.

- pfaults: (macOS): number of page faults.

- pageins: (macOS): number of actual pageins.

For on explanation of Windows fields see the PROCESS_MEMORY_COUNTERS_EX structure.

Throws a zombie_process() error for zombie processes.

Value

Named real vector.

Examples

```
p <- ps_handle()
p
ps_memory_info(p)
```

See Also

Other process handle functions: ps_children, ps_cmdline, ps_cpu_times, ps_create_time,
ps_cwd, ps_environ, ps_exe, ps_handle, ps_is_running, ps_kill, ps_name, ps_num_threads,
ps_pid, ps_ppid, ps_resume, ps_send_signal, ps_status, ps_suspend, ps_terminal, ps_terminate,
ps_uids, ps_username

---

ps_name   *Process name*

Description

The name of the program, which is typically the name of the executable.

Usage

```
ps_name(p)
```

Arguments

p   Process handle.

Details

On on Unix this can change, e.g. via an exec*() system call.

ps_name() works on zombie processes.

Value

Character scalar.

Examples

```
p <- ps_handle()
p
ps_name(p)
ps_exe(p)
ps_cmdline(p)
```

See Also

Other process handle functions: ps_children, ps_cmdline, ps_cpu_times, ps_create_time,
ps_cwd, ps_environ, ps_exe, ps_handle, ps_is_running, ps_kill, ps_memory_info, ps_num_threads,
ps_pid, ps_ppid, ps_resume, ps_send_signal, ps_status, ps_suspend, ps_terminal, ps_terminate,
ps_uids, ps_username

---

ps_num_threads   *Number of threads*

Description

Throws a `zombie_process()` error for zombie processes.

Usage

```
ps_num_threads(p)
```

Arguments

p   Process handle.

Value

Integer scalar.

Examples

```
p <- ps_handle()
p
ps_num_threads(p)
```

See Also

Other process handle functions: `ps_children`, `ps_cmdline`, `ps_cpu_times`, `ps_create_time`, `ps_cwd`, `ps_environ`, `ps_exe`, `ps_handle`, `ps_is_running`, `ps_kill`, `ps_memory_info`, `ps_name`, `ps_pid`, `ps_ppid`, `ps_resume`, `ps_send_signal`, `ps_status`, `ps_suspend`, `ps_terminal`, `ps_terminate`, `ps_uids`, `ps_username`

---

ps_os_type   *Query the type of the OS*

Description

Query the type of the OS

Usage

```
ps_os_type()

ps_is_supported()
```

Value

ps_os_type returns a named logical vector. The rest of the functions return a logical scalar.

ps_is_supported() returns TRUE if ps supports the current platform.

Examples

```
ps_os_type()
ps_is_supported()
```

---

ps_pid   *Pid of a process handle*

Description

This function works even if the process has already finished.

Usage

```
ps_pid(p)
```

Arguments

p   Process handle.

Value

Process id.

Examples

```
p <- ps_handle()
p
ps_pid(p)
ps_pid(p) == Sys.getpid()
```

See Also

Other process handle functions: [ps_children](), [ps_cmdline](), [ps_cpu_times](), [ps_create_time](), [ps_cwd](), [ps_environ](), [ps_exe](), [ps_handle](), [ps_is_running](), [ps_kill](), [ps_memory_info](), [ps_name](), [ps_num_threads](), [ps_ppid](), [ps_resume](), [ps_send_signal](), [ps_status](), [ps_suspend](), [ps_terminal](), [ps_terminate](), [ps_uids](), [ps_username]()

---

ps_pids   *Ids of all processes on the system*

Description

Ids of all processes on the system

Usage

```
ps_pids()
```

Value

Integer vector of process ids.

---

ps_ppid *Parent pid or parent process of a process*

Description

ps_ppid() returns the parent pid, ps_parent() returns a ps_handle of the parent.

Usage

```
ps_ppid(p)

ps_parent(p)
```

Arguments

p   Process handle.

Details

On POSIX systems, if the parent process terminates, another process (typically the pid 1 process) is marked as parent. ps_ppid() and ps_parent() will return this process then.

Both ps_ppid() and ps_parent() work for zombie processes.

Value

ps_ppid() returns and integer scalar, the pid of the parent of p. ps_parent() returns a ps_handle.

Examples

```
p <- ps_handle()
p
ps_ppid(p)
ps_parent(p)
```

See Also

Other process handle functions: ps_children, ps_cmdline, ps_cpu_times, ps_create_time, ps_cwd, ps_environ, ps_exe, ps_handle, ps_is_running, ps_kill, ps_memory_info, ps_name, ps_num_threads, ps_pid, ps_resume, ps_send_signal, ps_status, ps_suspend, ps_terminal, ps_terminate, ps_uids, ps_username

Other process handle functions: ps_children, ps_cmdline, ps_cpu_times, ps_create_time, ps_cwd, ps_environ, ps_exe, ps_handle, ps_is_running, ps_kill, ps_memory_info, ps_name, ps_num_threads, ps_pid, ps_resume, ps_send_signal, ps_status, ps_suspend, ps_terminal, ps_terminate, ps_uids, ps_username

---

ps_resume   *Resume (continue) a stopped process*

Description

Resume process execution with SIGCONT pre-emptively checking whether PID has been reused. On Windows this has the effect of resuming all process threads.

Usage

```
ps_resume(p)
```

Arguments

p   Process handle.

Examples

```
px <- processx::process$new("sleep", "10")
p <- ps_handle(px$get_pid())
p
ps_suspend(p)
ps_status(p)
ps_resume(p)
ps_status(p)
ps_kill(p)
```

See Also

Other process handle functions: `ps_children`, `ps_cmdline`, `ps_cpu_times`, `ps_create_time`, `ps_cwd`, `ps_environ`, `ps_exe`, `ps_handle`, `ps_is_running`, `ps_kill`, `ps_memory_info`, `ps_name`, `ps_num_threads`, `ps_pid`, `ps_ppid`, `ps_send_signal`, `ps_status`, `ps_suspend`, `ps_terminal`, `ps_terminate`, `ps_uids`, `ps_username`

---

ps_send_signal   *Send signal to a process*

Description

Send a signal to the process. Not implemented on Windows. See `signals()` for the list of signals on the current platform.

Usage

```
ps_send_signal(p, sig)
```

Arguments

p   Process handle.

sig   Signal number, see `signals()`.

Details

It checks if the process is still running, before sending the signal, to avoid signalling the wrong process, because of pid reuse.

Examples

```
px <- processx::process$new("sleep", "10")
p <- ps_handle(px$get_pid())
p
ps_send_signal(p, signals()$SIGINT)
p
ps_is_running(p)
px$get_exit_status()
```

See Also

Other process handle functions: `ps_children`, `ps_cmdline`, `ps_cpu_times`, `ps_create_time`, `ps_cwd`, `ps_environ`, `ps_exe`, `ps_handle`, `ps_is_running`, `ps_kill`, `ps_memory_info`, `ps_name`, `ps_num_threads`, `ps_pid`, `ps_ppid`, `ps_resume`, `ps_status`, `ps_suspend`, `ps_terminal`, `ps_terminate`, `ps_uids`, `ps_username`

---

ps_status	*Current process status*

Description

One of the following:

- `"idle"`: Process being created by fork, macOS only.
- `"running"`: Currently runnable on macOS and Windows. Actually running on Linux.
- `"sleeping"` Sleeping on a wait or poll.
- `"disk_sleep"` Uninterruptible sleep, waiting for an I/O operation (Linux only).
- `"stopped"` Stopped, either by a job control signal or because it is being traced.
- `"tracing_stop"` Stopped for tracing (Linux only).
- `"zombie"` Zombie. Finished, but parent has not read out the exit status yet.
- `"dead"` Should never be seen (Linux).
- `"wake_kill"` Received fatal signal (Linux only).
- `"waking"` Paging (Linux only, not valid since the 2.6.xx kernel).

Usage

```
ps_status(p)
```

Arguments

p   Process handle.

Details

Works for zombie processes.

Value

Character scalar.

Examples

```
p <- ps_handle()
p
ps_status(p)
```

See Also

Other process handle functions: ps_children, ps_cmdline, ps_cpu_times, ps_create_time, ps_cwd, ps_environ, ps_exe, ps_handle, ps_is_running, ps_kill, ps_memory_info, ps_name, ps_num_threads, ps_pid, ps_ppid, ps_resume, ps_send_signal, ps_suspend, ps_terminal, ps_terminate, ps_uids, ps_username

---

ps_suspend   *Suspend (stop) the process*

Description

Suspend process execution with SIGSTOP pre-emptively checking whether PID has been reused. On Windows this has the effect of suspending all process threads.

Usage

```
ps_suspend(p)
```

Arguments

p   Process handle.

Examples

```
px <- processx::process$new("sleep", "10")
p <- ps_handle(px$get_pid())
p
ps_suspend(p)
ps_status(p)
ps_resume(p)
ps_status(p)
ps_kill(p)
```

See Also

Other process handle functions: ps_children, ps_cmdline, ps_cpu_times, ps_create_time, ps_cwd, ps_environ, ps_exe, ps_handle, ps_is_running, ps_kill, ps_memory_info, ps_name, ps_num_threads, ps_pid, ps_ppid, ps_resume, ps_send_signal, ps_status, ps_terminal, ps_terminate, ps_uids, ps_username

ps_terminal   *Terminal device of the process*

Description

Returns the terminal of the process. Not implemented on Windows, always returns `NA_character_`.
On Unix it returns `NA_character_` if the process has no terminal.

Usage

```
ps_terminal(p)
```

Arguments

p   Process handle.

Details

Works for zombie processes.

Value

Character scalar.

Examples

```
p <- ps_handle()
p
ps_terminal(p)
```

See Also

Other process handle functions: `ps_children`, `ps_cmdline`, `ps_cpu_times`, `ps_create_time`,
`ps_cwd`, `ps_environ`, `ps_exe`, `ps_handle`, `ps_is_running`, `ps_kill`, `ps_memory_info`, `ps_name`,
`ps_num_threads`, `ps_pid`, `ps_ppid`, `ps_resume`, `ps_send_signal`, `ps_status`, `ps_suspend`, `ps_terminate`,
`ps_uids`, `ps_username`

ps_terminate   *Terminate a Unix process*

Description

Send a `SIGTERM` signal to the process. Not implemented on Windows.

Usage

```
ps_terminate(p)
```

Arguments

p   Process handle.

Details

Checks if the process is still running, to work around pid reuse.

Examples

```
px <- processx::process$new("sleep", "10")
p <- ps_handle(px$get_pid())
p
ps_terminate(p)
p
ps_is_running(p)
px$get_exit_status()
```

See Also

Other process handle functions: ps_children, ps_cmdline, ps_cpu_times, ps_create_time, ps_cwd, ps_environ, ps_exe, ps_handle, ps_is_running, ps_kill, ps_memory_info, ps_name, ps_num_threads, ps_pid, ps_ppid, ps_resume, ps_send_signal, ps_status, ps_suspend, ps_terminal, ps_uids, ps_username

---

ps_uids *User ids and group ids of the process*

Description

User ids and group ids of the process. Both return integer vectors with names: real, effective and saved.

Usage

```
ps_uids(p)

ps_gids(p)
```

Arguments

p   Process handle.

Details

Both work for zombie processes.

They are not implemented on Windows, they throw a not_implemented error.

Value

Named integer vector of length 3, with names: real, effective and saved.

Examples

```
p <- ps_handle()
p
ps_uids(p)
ps_gids(p)
```

See Also

ps_username() returns a user *name* and works on all platforms.

Other process handle functions: ps_children, ps_cmdline, ps_cpu_times, ps_create_time, ps_cwd, ps_environ, ps_exe, ps_handle, ps_is_running, ps_kill, ps_memory_info, ps_name, ps_num_threads, ps_pid, ps_ppid, ps_resume, ps_send_signal, ps_status, ps_suspend, ps_terminal, ps_terminate, ps_username

Other process handle functions: ps_children, ps_cmdline, ps_cpu_times, ps_create_time, ps_cwd, ps_environ, ps_exe, ps_handle, ps_is_running, ps_kill, ps_memory_info, ps_name, ps_num_threads, ps_pid, ps_ppid, ps_resume, ps_send_signal, ps_status, ps_suspend, ps_terminal, ps_terminate, ps_username

---

ps_username   *Owner of the process*

Description

The name of the user that owns the process. On Unix it is calculated from the real user id.

Usage

```
ps_username(p)
```

Arguments

p   Process handle.

Details

On Unix, a numeric uid id returned if the uid is not in the user database, thus a username cannot be determined.

Works for zombie processes.

Value

String scalar.

Examples

```
p <- ps_handle()
p
ps_username(p)
```

See Also

Other process handle functions: ps_children, ps_cmdline, ps_cpu_times, ps_create_time, ps_cwd, ps_environ, ps_exe, ps_handle, ps_is_running, ps_kill, ps_memory_info, ps_name, ps_num_threads, ps_pid, ps_ppid, ps_resume, ps_send_signal, ps_status, ps_suspend, ps_terminal, ps_terminate, ps_uids

signals	*List of all supported signals*

Description

Only the signals supported by the current platform are included.

Usage

```
signals()
```

Value

List of integers, named by signal names.

Index