

**NAME**

**procstat\_close**, **procstat\_freeadvlock**, **procstat\_freeargv**, **procstat\_freeauxv**, **procstat\_freenvv**, **procstat\_freefiles**, **procstat\_freegroups**, **procstat\_freekstack**, **procstat\_freeprocs**, **procstat\_freeptlwpinfo**, **procstat\_freevmmap**, **procstat\_get\_pipe\_info**, **procstat\_get\_pts\_info**, **procstat\_get\_sem\_info**, **procstat\_get\_shm\_info**, **procstat\_get\_socket\_info**, **procstat\_get\_vnode\_info**, **procstat\_getadvlock**, **procstat\_getargv**, **procstat\_getauxv**, **procstat\_getenvv**, **procstat\_getfiles**, **procstat\_getgroups**, **procstat\_getkstack**, **procstat\_getosrel**, **procstat\_getpathname**, **procstat\_getprocs**, **procstat\_getptlwpinfo**, **procstat\_getrlimit**, **procstat\_getumask**, **procstat\_getvmmap**, **procstat\_open\_core**, **procstat\_open\_kvm**, **procstat\_open\_sysctl** - library interface for file and process information retrieval

**LIBRARY**

Process and Files Information Retrieval (libprocstat, -lprocstat)

**SYNOPSIS**

```
#include <sys/param.h>
```

```
#include <sys/queue.h>
```

```
#include <sys/socket.h>
```

```
#include <libprocstat.h>
```

*void*

```
procstat_close(struct procstat *procstat);
```

*void*

```
procstat_freeadvlock(struct procstat *procstat, struct advlock_list *list);
```

*void*

```
procstat_freeargv(struct procstat *procstat);
```

*void*

```
procstat_freeauxv(struct procstat *procstat, Elf_Auxinfo *auxv);
```

*void*

```
procstat_freenvv(struct procstat *procstat);
```

*void*

```
procstat_freefiles(struct procstat *procstat, struct filestat_list *head);
```

*void*

```
procstat_freegroups(struct procstat *procstat, gid_t *groups);
```

*void*

**procstat\_freekstack**(*struct procstat \*procstat, struct kinfo\_kstack \*kkstp*);

*void*

**procstat\_freeprocs**(*struct procstat \*procstat, struct kinfo\_proc \*p*);

*void*

**procstat\_freevmmap**(*struct procstat \*procstat, struct kinfo\_vmentry \*vmmap*);

*void*

**procstat\_freeptrace\_lwpinfo**(*struct procstat \*procstat, struct ptrace\_lwpinfo \*pl*);

*int*

**procstat\_get\_pipe\_info**(*struct procstat \*procstat, struct filestat \*fst, struct pipestat \*pipe, char \*errbuf*);

*int*

**procstat\_get\_pts\_info**(*struct procstat \*procstat, struct filestat \*fst, struct ptsstat \*pts, char \*errbuf*);

*int*

**procstat\_get\_sem\_info**(*struct procstat \*procstat, struct filestat \*fst, struct semstat \*sem, char \*errbuf*);

*int*

**procstat\_get\_shm\_info**(*struct procstat \*procstat, struct filestat \*fst, struct shmstat \*shm, char \*errbuf*);

*int*

**procstat\_get\_socket\_info**(*struct procstat \*procstat, struct filestat \*fst, struct sockstat \*sock, char \*errbuf*);

*int*

**procstat\_get\_vnode\_info**(*struct procstat \*procstat, struct filestat \*fst, struct vnstat \*vn, char \*errbuf*);

*struct advlock\_list \**

**procstat\_getadvlock**(*struct procstat \*procstat*);

*char \*\**

**procstat\_getargv**(*struct procstat \*procstat, const struct kinfo\_proc \*kp, size\_t nchr*);

*Elf\_Auxinfo \**

**procstat\_getauxv**(*struct procstat \*procstat, struct kinfo\_proc \*kp, unsigned int \*count*);

*char \*\**

**procstat\_getenvv**(*struct procstat \*procstat, const struct kinfo\_proc \*kp, size\_t nchr*);

*struct filestat\_list \**

**procstat\_getfiles**(*struct procstat \*procstat, struct kinfo\_proc \*kp, int mmapped*);

*gid\_t \**

**procstat\_getgroups**(*struct procstat \*procstat, struct kinfo\_proc \*kp, unsigned int \*count*);

*struct kinfo\_kstack \**

**procstat\_getkstack**(*struct procstat \*procstat, struct kinfo\_proc \*kp, unsigned int \*count*);

*int*

**procstat\_getosrel**(*struct procstat \*procstat, struct kinfo\_proc \*kp, int \*osrelp*);

*int*

**procstat\_getpathname**(*struct procstat \*procstat, struct kinfo\_proc \*kp, char \*pathname, size\_t maxlen*);

*struct kinfo\_proc \**

**procstat\_getprocs**(*struct procstat \*procstat, int what, int arg, unsigned int \*count*);

*struct ptrace\_lwpinfo \**

**procstat\_getptlwpinfo**(*struct procstat \*procstat, unsigned int \*count*);

*int*

**procstat\_getrlimit**(*struct procstat \*procstat, struct kinfo\_proc \*kp, int which, struct rlimit\* rlimit*);

*int*

**procstat\_getumask**(*struct procstat \*procstat, struct kinfo\_proc \*kp, unsigned short \*maskp*);

*struct kinfo\_vmentry \**

**procstat\_getvmmmap**(*struct procstat \*procstat, struct kinfo\_proc \*kp, unsigned int \*count*);

*struct procstat \**

**procstat\_open\_core**(*const char \*filename*);

*struct procstat \**

**procstat\_open\_kvm**(*const char \*nlistf, const char \*memf*);

*struct procstat \**

**procstat\_open\_sysctl**(void);

## DESCRIPTION

The **libprocstat** library contains the API for runtime file and process information retrieval from the running kernel via the **sysctl(3)** library backend, and for post-mortem analysis via the **kvm(3)** library backend, or from the process **core(5)** file, searching for statistics in special **elf(3)** note sections.

The **procstat\_open\_kvm()** and **procstat\_open\_sysctl()** functions use the **kvm(3)** or **sysctl(3)** library routines, respectively, to access kernel state information used to retrieve processes and files states. The **procstat\_open\_core()** uses **elf(3)** routines to access statistics stored as a set of notes in a process **core(5)** file, written by the kernel at the moment of the process abnormal termination. The *filename* argument is the process core file name. The *nlistf* argument is the executable image of the kernel being examined. If this argument is **NULL**, the currently running kernel is assumed. The *memf* argument is the kernel memory device file. If this argument is **NULL**, then */dev/mem* is assumed. See **kvm\_open(3)** for more details. The functions dynamically allocate and return a *procstat* structure pointer used in the rest of the **libprocstat** library routines until the corresponding **procstat\_close()** call that cleans up the resources allocated by the **procstat\_open\_\***(*\**) functions.

The **procstat\_getprocs()** function gets a pointer to the *procstat* structure from one of the **procstat\_open\_\***(*\**) functions and returns a dynamically allocated (sub-)set of active processes in the kernel filled in to array of *kinfo\_proc* structures. The *what* and *arg* arguments constitute a filtering predicate as described in the **kvm\_getprocs(3)** function. The number of processes found is returned in the reference parameter *cnt*. The caller is responsible to free the allocated memory with a subsequent **procstat\_freeprocs()** function call.

The **procstat\_getptlwpinfo()** function gets a pointer to the *procstat* structure from the **procstat\_open\_core()** function and returns a dynamically allocated set of signals intercepted by a process in the process's core file. The number of processes found is returned in the reference parameter *cnt*. The caller is responsible to free the allocated memory with a subsequent **procstat\_freeptlwpinfo()** function call.

The **procstat\_getargv()** function gets a pointer to the *procstat* structure from one of the **procstat\_open\_\***(*\**) functions, a pointer to *kinfo\_proc* structure from the array obtained from the **procstat\_getprocs()** function, and returns a null-terminated argument vector that corresponds to the command line arguments passed to the process. The *nchr* argument indicates the maximum number of characters, including null bytes, to use in building the strings. If this amount is exceeded, the string causing the overflow is truncated and the partial result is returned. This is handy for programs that print only a one line summary of a command and should not copy out large amounts of text only to ignore it. If *nchr* is zero, no limit is imposed and all argument strings are returned. The values of the returned argument vector refer the strings stored in the *procstat* internal buffer. A subsequent call of the function with the same

*procstat* argument will reuse the buffer. To free the allocated memory **procstat\_freeargv()** function call can be used, or it will be released on **procstat\_close()**.

The **procstat\_getenvv()** function is similar to **procstat\_getargv()** but returns the vector of environment strings. The caller may free the allocated memory with a subsequent **procstat\_freeenvv()** function call.

The **procstat\_getauxv()** function gets a pointer to the *procstat* structure, a pointer to *kinfo\_proc* structure, and returns the auxiliary vector as a dynamically allocated array of *Elf\_Auxinfo* elements. The caller is responsible to free the allocated memory with a subsequent **procstat\_freeauxv()** function call.

The **procstat\_getfiles()** function gets a pointer to the *procstat* structure initialized with one of the **procstat\_open\_\***() functions, a pointer to *kinfo\_proc* structure from the array obtained from the **procstat\_getprocs()** function, and returns a dynamically allocated linked list of filled in *filestat\_list* structures using the STAILQ macros defined in *queue(3)*. The caller is responsible to free the allocated memory with a subsequent **procstat\_freefiles()** function call.

The **procstat\_getgroups()** function gets a pointer to the *procstat* structure, a pointer to *kinfo\_proc* structure, and returns the process groups as a dynamically allocated array of *gid\_t* elements. The caller is responsible to free the allocated memory with a subsequent **procstat\_freegrups()** function call.

The **procstat\_getkstack()** function gets a pointer to the *procstat* structure initialized with one of the **procstat\_open\_\***() functions, a pointer to *kinfo\_proc* structure, and returns kernel stacks of the process as a dynamically allocated array of *kinfo\_kstack* structures. The caller is responsible to free the allocated memory with a subsequent **procstat\_freekstack()** function call.

The **procstat\_getosrel()** function gets a pointer to the *procstat* structure, a pointer to *kinfo\_proc* structure, and returns osrel date in the 3rd reference parameter.

The **procstat\_getpathname()** function gets a pointer to the *procstat* structure, a pointer to *kinfo\_proc* structure, and copies the path of the process executable to *pathname* buffer, limiting to *maxlen* characters.

The **procstat\_getrlimit()** function gets a pointer to the *procstat* structure, a pointer to *kinfo\_proc* structure, resource index *which*, and returns the actual resource limit in the 4th reference parameter.

The **procstat\_getumask()** function gets a pointer to the *procstat* structure, a pointer to *kinfo\_proc* structure, and returns the process umask in the 3rd reference parameter.

The **procstat\_getvmmap()** function gets a pointer to the *procstat* structure initialized with one of the **procstat\_open\_\***() functions, a pointer to *kinfo\_proc* structure, and returns VM layout of the process as a

dynamically allocated array of *kinfo\_vmentry* structures. The caller is responsible to free the allocated memory with a subsequent **procstat\_freevmmap()** function call.

The **procstat\_getadvlock()** function returns a dynamically allocated list of *struct advlock* structures, providing a snapshot of the currently acquired advisory locks in the system. Both locally acquired POSIX ( *fcntl(2)*) and BSD-style ( *flock(2)*) locks are reported, as well as locks established by remote file system protocols. For each lock, unique identifiers for the locked file and its mount point are guaranteed to be provided. If a path for the locked file can be reconstructed, it is provided as well. The returned list must be freed with the **procstat\_freeadvlock()** function.

The **procstat\_get\_pipe\_info()**, **procstat\_get\_pts\_info()**, **procstat\_get\_sem\_info()**, **procstat\_get\_shm\_info()**, **procstat\_get\_socket\_info()** and **procstat\_get\_vnode\_info()** functions are used to retrieve information about pipes, pseudo-terminals, semaphores, shared memory objects, sockets, and vnodes, respectively. Each of them have a similar interface API. The *procstat* argument is a pointer obtained from one of **procstat\_open\_\***() functions. The *filestat fst* argument is an element of STAILQ linked list as obtained from the **procstat\_getfiles()** function. The *filestat* structure contains a *fs\_type* field that specifies a file type and a corresponding function to be called among the **procstat\_get\_\*\_info** function family. The actual object is returned in the 3rd reference parameter. The *errbuf* argument indicates an actual error message in case of failure.

PS_FST_TYPE_FIFO	<b>procstat_get_vnode_info</b>
PS_FST_TYPE_VNODE	
	<b>procstat_get_vnode_info</b>
PS_FST_TYPE_SOCKET	
	<b>procstat_get_socket_info</b>
PS_FST_TYPE_PIPE	<b>procstat_get_pipe_info</b>
PS_FST_TYPE_PTS	<b>procstat_get_pts_info</b>
PS_FST_TYPE_SEM	<b>procstat_get_sem_info</b>
PS_FST_TYPE_SHM	<b>procstat_get_shm_info</b>

## SEE ALSO

*fstat(1)*, *fuser(1)*, *pipe(2)*, *shm\_open(2)*, *socket(2)*, *elf(3)*, *kvm(3)*, *queue(3)*, *sem\_open(3)*, *sysctl(3)*, *pts(4)*, *core(5)*, *vnode(9)*

## HISTORY

The **libprocstat** library appeared in FreeBSD 9.0.

## AUTHORS

The **libprocstat** library was written by Stanislav Sedov <*stas@FreeBSD.org*>.

This manual page was written by Sergey Kandaurov <[pluknet@FreeBSD.org](mailto:pluknet@FreeBSD.org)>.