NAME

procstat_close, **procstat_freeadvlock**, **procstat_freeargv**, **procstat_freeauxv**, **procstat_freeenvv**, **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_freeenvv(*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_freeptlwpinfo(*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_getvmmap(*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_freegroups**() 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	**procstat_get_vnode_info**
PS_FST_TYPE_VNODE	**procstat_get_vnode_info**
PS_FST_TYPE_SOCKET	**procstat_get_socket_info**
PS_FST_TYPE_PIPE	**procstat_get_pipe_info**
PS_FST_TYPE_PTS	**procstat_get_pts_info**
PS_FST_TYPE_SEM	**procstat_get_sem_info**
PS_FST_TYPE_SHM	**procstat_get_shm_info**

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*>.