Rocky Enterprise Linux 9.2 Manual Pages on command 'sem_overview.7'

$ man sem_overview.7

SEM_OVERVIEW(7) Linux Programmer's Manual SEM_OVERVIEW(7)

NAME

sem_overview - overview of POSIX semaphores

DESCRIPTION

POSIX semaphores allow processes and threads to synchronize their ac?

tions.

A semaphore is an integer whose value is never allowed to fall below

zero. Two operations can be performed on semaphores: increment the

semaphore value by one (sem_post(3)); and decrement the semaphore value

by one (sem_wait(3)). If the value of a semaphore is currently zero,

then a sem_wait(3) operation will block until the value becomes greater

than zero.

POSIX semaphores come in two forms: named semaphores and unnamed sema?

phores.

Named semaphores

A named semaphore is identified by a name of the form /somename;

that is, a null-terminated string of up to NAME_MAX-4 (i.e.,

251) characters consisting of an initial slash, followed by one

or more characters, none of which are slashes. Two processes can operate on the same named semaphore by passing the same name to sem_open(3).

The sem_open(3) function creates a new named semaphore or opens an existing named semaphore. After the semaphore has been opened, it can be operated on using sem_post(3) and sem_wait(3). When a process has finished using the semaphore, it can use sem_close(3) to close the semaphore. When all processes have finished using the semaphore, it can be removed from the system using sem_unlink(3).

Unnamed semaphores (memory-based semaphores)

An unnamed semaphore does not have a name. Instead the sema? phore is placed in a region of memory that is shared between multiple threads (a thread-shared semaphore) or processes (a process-shared semaphore). A thread-shared semaphore is placed in an area of memory shared between the threads of a process, for example, a global variable. A process-shared semaphore must be placed in a shared memory region (e.g., a System V shared memory segment created using shmget(2), or a POSIX shared memory object built created using shm_open(3)).

Before being used, an unnamed semaphore must be initialized us? ing sem_init(3). It can then be operated on using sem_post(3) and sem_wait(3). When the semaphore is no longer required, and before the memory in which it is located is deallocated, the semaphore should be destroyed using sem_destroy(3).

The remainder of this section describes some specific details of the Linux implementation of POSIX semaphores.

Versions

Prior to kernel 2.6, Linux supported only unnamed, thread-shared sema? phores. On a system with Linux 2.6 and a glibc that provides the NPTL threading implementation, a complete implementation of POSIX semaphores is provided.

Persistence

POSIX named semaphores have kernel persistence: if not removed by sem_unlink(3), a semaphore will exist until the system is shut down.

Linking

Programs using the POSIX semaphores API must be compiled with cc -pthread to link against the real-time library, librt.

Accessing named semaphores via the filesystem

On Linux, named semaphores are created in a virtual filesystem, normally mounted under /dev/shm, with names of the form sem.somename. (This is the reason that semaphore names are limited to NAME_MAX-4 rather than NAME_MAX characters.)

Since Linux 2.6.19, ACLs can be placed on files under this directory, to control object permissions on a per-user and per-group basis.

NOTES

System V semaphores (semget(2), semop(2), etc.) are an older semaphore API. POSIX semaphores provide a simpler, and better designed interface than System V semaphores; on the other hand POSIX semaphores are less widely available (especially on older systems) than System V semaphores.

EXAMPLES

An example of the use of various POSIX semaphore functions is shown in sem_wait(3).

SEE ALSO

sem_close(3), sem_destroy(3), sem_getvalue(3), sem_init(3), sem_open(3), sem_post(3), sem_unlink(3), sem_wait(3), pthreads(7), shm_overview(7)

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/.