Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'sem_open.3p' command

$ man sem_open.3p

SEM_OPEN(3P) POSIX Programmer's Manual SEM_OPEN(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

sem_open ? initialize and open a named semaphore

SYNOPSIS

#include <semaphore.h>

sem_t *sem_open(const char *name, int oflag, ...);

DESCRIPTION

The sem_open() function shall establish a connection between a named semaphore and a process. Following a call to sem_open() with semaphore name name, the process may reference the semaphore associated with name using the address returned from the call. This semaphore may be used in subsequent calls to sem_wait(), sem_timedwait(), sem_trywait(), sem_post(), and sem_close(). The semaphore remains usable by this process until the semaphore is closed by a successful call to sem_close(), _exit(), or one of the exec functions.

The oflag argument controls whether the semaphore is created or merely accessed by the call to sem_open(). The following flag bits may be set in oflag:

O_CREAT This flag is used to create a semaphore if it does not al?
ready exist. If O_CREAT is set and the semaphore already ex?
ists, then O_CREAT has no effect, except as noted under
O_EXCL. Otherwise, sem_open() creates a named semaphore. The
O_CREAT flag requires a third and a fourth argument: mode,
which is of type mode_t, and value, which is of type un?
signed. The semaphore is created with an initial value of
value. Valid initial values for semaphores are less than or
equal to {SEM_VALUE_MAX}.
The user ID of the semaphore shall be set to the effective
user ID of the process. The group ID of the semaphore shall
be set to the effective group ID of the process; however, if
the name argument is visible in the file system, the group ID
may be set to the group ID of the containing directory. The
permission bits of the semaphore are set to the value of the
mode argument except those set in the file mode creation mask
of the process. When bits in mode other than the file permis?
sion bits are specified, the effect is unspecified.
After the semaphore named name has been created by sem_open()
with the O_CREAT flag, other processes can connect to the
semaphore by calling sem_open() with the same value of name.
O_EXCL If O_EXCL and O_CREAT are set, sem_open() fails if the sema?
phore name exists. The check for the existence of the sema?
phore and the creation of the semaphore if it does not exist
are atomic with respect to other processes executing
sem_open() with O_EXCL and O_CREAT set. If O_EXCL is set and
O_CREAT is not set, the effect is undefined.
If flags other than O_CREAT and O_EXCL are specified in the
oflag parameter, the effect is unspecified.
The name argument points to a string naming a semaphore object. It is
unspecified whether the name appears in the file system and is visible
to functions that take pathnames as arguments. The name argument con?
forms to the construction rules for a pathname, except that the inter?

pretation of <slash> characters other than the leading <slash> charac?
ter in name is implementation-defined, and that the length limits for
the name argument are implementation-defined and need not be the same
as the pathname limits {PATH_MAX} and {NAME_MAX}. If name begins with
the <slash> character, then processes calling sem_open() with the same
value of name shall refer to the same semaphore object, as long as that
name has not been removed. If name does not begin with the <slash>
character, the effect is implementation-defined.

If a process makes multiple successful calls to sem_open() with the
same value for name, the same semaphore address shall be returned for
each such successful call, provided that there have been no calls to
sem_unlink() for this semaphore, and at least one previous successful
sem_open() call for this semaphore has not been matched with a
sem_close() call.

References to copies of the semaphore produce undefined results.

RETURN VALUE

Upon successful completion, the sem_open() function shall return the
address of the semaphore. Otherwise, it shall return a value of
SEM_FAILED and set errno to indicate the error. The symbol SEM_FAILED
is defined in the <semaphore.h> header. No successful return from
sem_open() shall return the value SEM_FAILED.

ERRORS

If any of the following conditions occur, the sem_open() function shall
return SEM_FAILED and set errno to the corresponding value:

EACCES The named semaphore exists and the permissions specified by
oflag are denied, or the named semaphore does not exist and per?
mission to create the named semaphore is denied.

EEXIST O_CREAT and O_EXCL are set and the named semaphore already ex?
ists.

EINTR The sem_open() operation was interrupted by a signal.

EINVAL The sem_open() operation is not supported for the given name, or
O_CREAT was specified in oflag and value was greater than
{SEM_VALUE_MAX}.

EMFILE Too many semaphore descriptors or file descriptors are currently in use by this process.

ENFILE Too many semaphores are currently open in the system.

ENOENT O_CREAT is not set and the named semaphore does not exist.

ENOMEM There is insufficient memory for the creation of the new named semaphore.

ENOSPC There is insufficient space on a storage device for the creation of the new named semaphore.

If any of the following conditions occur, the sem_open() function may return SEM_FAILED and set errno to the corresponding value:

ENAMETOOLONG

The length of the name argument exceeds {_POSIX_PATH_MAX} on systems that do not support the XSI option or exceeds {_XOPEN_PATH_MAX} on XSI systems, or has a pathname component that is longer than {_POSIX_NAME_MAX} on systems that do not support the XSI option or longer than {_XOPEN_NAME_MAX} on XSI systems.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

Early drafts required an error return value of -1 with the type sem_t * for the sem_open() function, which is not guaranteed to be portable across implementations. The revised text provides the symbolic error code SEM_FAILED to eliminate the type conflict.

FUTURE DIRECTIONS

A future version might require the sem_open() and sem_unlink() functions to have semantics similar to normal file system operations.

SEE ALSO

semctl(), semget(), semop(), sem_close(), sem_post(), sem_timedwait(), sem_trywait(), sem_unlink()

The Base Definitions volume of POSIX.1?2017, <semaphore.h>

COPYRIGHT