Rocky Enterprise Linux 9.2 Manual Pages on command 'shmctl.2'

$ man shmctl.2

SHMCTL(2) Linux Programmer's Manual SHMCTL(2)

NAME

shmctl - System V shared memory control

SYNOPSIS

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

#include <sys/shm.h>

int shmctl(int shmid, int cmd, struct shmid_ds *buf);
```

DESCRIPTION

shmctl() performs the control operation specified by cmd on the System V shared memory segment whose identifier is given in shmid.

The buf argument is a pointer to a shmid_ds structure, defined in <sys/shm.h> as follows:

```
struct shmid_ds {
    struct ipc_perm shm_perm;    /* Ownership and permissions */
    size_t          shm_segsz;   /* Size of segment (bytes) */
    time_t          shm_atime;   /* Last attach time */
    time_t          shm_dtime;   /* Last detach time */
    time_t          shm_ctime;   /* Creation time/time of last
                                    modification via shmctl() */
    pid_t           shm_cpid;    /* PID of creator */
    pid_t           shm_lpid;    /* PID of last shmat(2)/shmdt(2) */
    shmatt_t        shm_nattch;  /* No. of current attaches */
    ...
};
```

The fields of the shmid_ds structure are as follows:

shm_perm   This is an ipc_perm structure (see below) that specifies the access permis?
sions on the shared memory segment.

shm_segsz   Size in bytes of the shared memory segment.

shm_atime   Time of the last shmat(2) system call that attached this segment.

shm_dtime   Time of the last shmdt(2) system call that detached tgis segment.

shm_ctime   Time of creation of segment or time of the last shmctl() IPC_SET operation.

shm_cpid   ID of the process that created the shared memory segment.

shm_lpid   ID of the last process that executed a shmat(2) or shmdt(2) system call on
this segment.

shm_nattch Number of processes that have this segment attached.

The ipc_perm structure is defined as follows (the highlighted fields are settable using IPC_SET):

```
    struct ipc_perm {
        key_t          __key;    /* Key supplied to shmget(2) */
        uid_t          uid;      /* Effective UID of owner */
        gid_t          gid;      /* Effective GID of owner */
        uid_t          cuid;     /* Effective UID of creator */
        gid_t          cgid;     /* Effective GID of creator */
        unsigned short mode;     /* Permissions + SHM_DEST and
                                    SHM_LOCKED flags */
        unsigned short __seq;    /* Sequence number */
    };
```

The least significant 9 bits of the mode field of the ipc_perm structure define the access permissions for the shared memory segment. The permission bits are as follows:

0400	Read by user
0200	Write by user
0040	Read by group
0020	Write by group
0004	Read by others
0002	Write by others

Bits 0100, 0010, and 0001 (the execute bits) are unused by the system. (It is not neces?
sary to have execute permission on a segment in order to perform a shmat(2) call with the

SHM_EXEC flag.)

Valid values for cmd are:

IPC_STAT

Copy information from the kernel data structure associated with shmid into the shmid_ds structure pointed to by buf. The caller must have read permission on the shared memory segment.

IPC_SET

Write the values of some members of the shmid_ds structure pointed to by buf to the kernel data structure associated with this shared memory segment, updating also its shm_ctime member.

The following fields are updated: shm_perm.uid, shm_perm.gid, and (the least sig?nificant 9 bits of) shm_perm.mode.

The effective UID of the calling process must match the owner (shm_perm.uid) or creator (shm_perm.cuid) of the shared memory segment, or the caller must be privi?leged.

IPC_RMID

Mark the segment to be destroyed. The segment will actually be destroyed only af?ter the last process detaches it (i.e., when the shm_nattch member of the associ?ated structure shmid_ds is zero). The caller must be the owner or creator of the segment, or be privileged. The buf argument is ignored.

If a segment has been marked for destruction, then the (nonstandard) SHM_DEST flag of the shm_perm.mode field in the associated data structure retrieved by IPC_STAT will be set.

The caller must ensure that a segment is eventually destroyed; otherwise its pages that were faulted in will remain in memory or swap.

See also the description of /proc/sys/kernel/shm_rmid_forced in proc(5).

IPC_INFO (Linux-specific)

Return information about system-wide shared memory limits and parameters in the structure pointed to by buf. This structure is of type shminfo (thus, a cast is required), defined in <sys/shm.h> if the _GNU_SOURCE feature test macro is defined:

```
struct shminfo {
    unsigned long shmmax; /* Maximum segment size */
    unsigned long shmmin; /* Minimum segment size;
```

```
                                always 1 */
            unsigned long shmmni; /* Maximum number of segments */
            unsigned long shmseg; /* Maximum number of segments
                                that a process can attach;
                                unused within kernel */
            unsigned long shmall; /* Maximum number of pages of
                                shared memory, system-wide */
        };
```

The shmmni, shmmax, and shmall settings can be changed via /proc files of the same name; see proc(5) for details.

SHM_INFO (Linux-specific)

Return a shm_info structure whose fields contain information about system resources consumed by shared memory. This structure is defined in <sys/shm.h> if the _GNU_SOURCE feature test macro is defined:

```
        struct shm_info {
            int           used_ids; /* # of currently existing
                                 segments */
            unsigned long shm_tot;  /* Total number of shared
                                 memory pages */
            unsigned long shm_rss;  /* # of resident shared
                                 memory pages */
            unsigned long shm_swp;  /* # of swapped shared
                                 memory pages */
            unsigned long swap_attempts;
                               /* Unused since Linux 2.4 */
            unsigned long swap_successes;
                               /* Unused since Linux 2.4 */
        };
```

SHM_STAT (Linux-specific)

Return a shmid_ds structure as for IPC_STAT. However, the shmid argument is not a segment identifier, but instead an index into the kernel's internal array that maintains information about all shared memory segments on the system.

SHM_STAT_ANY (Linux-specific, since Linux 4.17)

Return a shmid_ds structure as for SHM_STAT. However, shm_perm.mode is not checked for read access for shmid, meaning that any user can employ this operation (just as any user may read /proc/sysvipc/shm to obtain the same information).

The caller can prevent or allow swapping of a shared memory segment with the following cmd values:

SHM_LOCK (Linux-specific)

Prevent swapping of the shared memory segment. The caller must fault in any pages that are required to be present after locking is enabled. If a segment has been locked, then the (nonstandard) SHM_LOCKED flag of the shm_perm.mode field in the associated data structure retrieved by IPC_STAT will be set.

SHM_UNLOCK (Linux-specific)

Unlock the segment, allowing it to be swapped out.

In kernels before 2.6.10, only a privileged process could employ SHM_LOCK and SHM_UNLOCK. Since kernel 2.6.10, an unprivileged process can employ these operations if its effective UID matches the owner or creator UID of the segment, and (for SHM_LOCK) the amount of mem? ory to be locked falls within the RLIMIT_MEMLOCK resource limit (see setrlimit(2)).

RETURN VALUE

A successful IPC_INFO or SHM_INFO operation returns the index of the highest used entry in the kernel's internal array recording information about all shared memory segments. (This information can be used with repeated SHM_STAT or SHM_STAT_ANY operations to obtain infor? mation about all shared memory segments on the system.) A successful SHM_STAT operation returns the identifier of the shared memory segment whose index was given in shmid. Other operations return 0 on success.

On error, -1 is returned, and errno is set appropriately.

ERRORS

EACCES IPC_STAT or SHM_STAT is requested and shm_perm.mode does not allow read access for shmid, and the calling process does not have the CAP_IPC_OWNER capability in the user namespace that governs its IPC namespace.

EFAULT The argument cmd has value IPC_SET or IPC_STAT but the address pointed to by buf isn't accessible.

EIDRM shmid points to a removed identifier.

EINVAL shmid is not a valid identifier, or cmd is not a valid command. Or: for a SHM_STAT or SHM_STAT_ANY operation, the index value specified in shmid referred to an array slot that is currently unused.

ENOMEM (In kernels since 2.6.9), SHM_LOCK was specified and the size of the to-be-locked segment would mean that the total bytes in locked shared memory segments would ex? ceed the limit for the real user ID of the calling process. This limit is defined by the RLIMIT_MEMLOCK soft resource limit (see setrlimit(2)).

EOVERFLOW

IPC_STAT is attempted, and the GID or UID value is too large to be stored in the structure pointed to by buf.

EPERM IPC_SET or IPC_RMID is attempted, and the effective user ID of the calling process is not that of the creator (found in shm_perm.cuid), or the owner (found in shm_perm.uid), and the process was not privileged (Linux: did not have the CAP_SYS_ADMIN capability).

Or (in kernels before 2.6.9), SHM_LOCK or SHM_UNLOCK was specified, but the process was not privileged (Linux: did not have the CAP_IPC_LOCK capability). (Since Linux 2.6.9, this error can also occur if the RLIMIT_MEMLOCK is 0 and the caller is not privileged.)

CONFORMING TO

POSIX.1-2001, POSIX.1-2008, SVr4.

NOTES

The inclusion of <sys/types.h> and <sys/ipc.h> isn't required on Linux or by any version of POSIX. However, some old implementations required the inclusion of these header files, and the SVID also documented their inclusion. Applications intended to be portable to such old systems may need to include these header files.

The IPC_INFO, SHM_STAT, and SHM_INFO operations are used by the ipcs(1) program to provide information on allocated resources. In the future, these may modified or moved to a /proc filesystem interface.

Linux permits a process to attach (shmat(2)) a shared memory segment that has already been marked for deletion using shmctl(IPC_RMID). This feature is not available on other UNIX implementations; portable applications should avoid relying on it.

Various fields in a struct shmid_ds were typed as short under Linux 2.2 and have become long under Linux 2.4. To take advantage of this, a recompilation under glibc-2.1.91 or later should suffice. (The kernel distinguishes old and new calls by an IPC_64 flag in cmd.)

SEE ALSO

mlock(2), setrlimit(2), shmget(2), shmop(2), capabilities(7), sysvipc(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/.