shmget

allocates a shared memory segment

Synopsis:-

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

int shmget(key_t *key***, size_t** *size***, int** *shmflg***);**

Description:-

shmget() returns the identifier of the shared memory segment associated with the value of the argument *key*. A new shared memory segment, with size equal to the value of *size* rounded up to a multiple of **PAGE_SIZE**, is created if *key* has the value **IPC_PRIVATE** or *key* isn't **IPC_PRIVATE**, no shared memory segment corresponding to *key* exists, and **IPC_CREAT** is specified in *shmflg*.

The value *shmflg* is composed of:

IPC_CREAT:-

to create a new segment. If this flag is not used, then **shmget**() will find the segment associated with *key* and check to see if the user has permission to access the segment.

IPC_EXCL:-

used with **IPC_CREAT** to ensure failure if the segment already exists.

mode_flags:-

(least significant 9 bits) specifying the permissions granted to the owner, group, and world. These bits have the same format, and the same meaning, as the *mode* argument of **open**(2). Presently, the execute permissions are not used by the system.

Return Value:-

A valid segment identifier, *shmid*, is returned on success, -1 on error.

For more information, use **man shmget**

shmat

void *shmat(int shmid, const void *shmaddr,int shmflg)

Description:-
shmat() attaches the shared memory segment identified by *shmid* to the address space of the calling process. The attaching address is specified by *shmaddr* with one of the following criteria:

If *shmaddr* is NULL, the system chooses a suitable (unused) address at which to attach the segment.

If *shmaddr* isn't NULL and **SHM_RND** is specified in *shmflg*, the attach occurs at the address equal to *shmaddr* rounded down to the nearest multiple of **SHMLBA**. Otherwise *shmaddr* must be a page-aligned address at which the attach occurs.

If **SHM_RDONLY** is specified in *shmflg*, the segment is attached for reading and the process must

have read permission for the segment. Otherwise the segment is attached for read and write and the process must have read and write permission for the segment. There is no notion of a write-only shared memory segment.

After a ***fork*** the child inherits the attached shared memory segments.

After an ***execve*** all attached shared memory segments are detached from the process.

Upon **exit** all attached shared memory segments are detached from the process.

Return Value:-

On success **shmat**() returns the address of the attached shared memory segment; on error *(void *) -1* is returned, and *errno* is set to indicate the cause of the error.

For more information, use **man shmat**

sigaction

examine and change a signal action

Synopsis:-

#include <signal.h>
int sigaction(int *signum***, const struct sigaction ****act***, struct sigaction ****oldact***);**

Description:-
The **sigaction**() system call is used to change the action taken by a process on receipt of a specific signal.
signum specifies the signal and can be any valid signal except **SIGKILL** and **SIGSTOP**.
If *act* is non-NULL, the new action for signal *signum* is installed from *act*. If *oldact* is non-NULL, the previous action is saved in *oldact*.
The *sigaction* structure is defined as something like:

```
struct sigaction {
    void (*sa_handler) (int);
    void (*sa_sigaction)(int, siginfo_t *, void *);
    sigset_t sa_mask;
    int sa_flags;
    void (*sa_restorer)(void);
};
```

Return Value:-

sigaction() returns 0 on success and -1 on error.

For more information, use **man sigaction**

kill

send a signal to a process or a group of processes

Synopsis:-
#include <signal.h>

int kill(pid_t *pid***, int** *sig***);**

Description:-
The *kill*() function shall send a signal to a process or a group of processes specified by *pid*. The signal to be sent is specified by *sig* and is either one from the list given in <**signal.h**> or 0. If *sig* is 0 (the null signal), error checking is performed but no signal is actually sent. The null signal can be used to check the validity of *pid*.

Return Value:-

Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to indicate the error.

For more information, use **man 2 kill**