**int shmget(key_t key, int size, int flags);**
Allocates a shared memory segment.

- key is the key associated with the shared memory segment you want.
- size is the size in bytes of the shared memory segment you want allocated. Memory gets allocated in pages, so chances are you'll probably get a little more memory than you wanted.
- flags indicate how you want the segment created and its access permissions. The general rule is just to use 0666 | IPC_CREAT | IPC_EXCL if the caller is making a new segment. If the caller wants to use an existing share region, simply pass 0 in the flag.

**Shmget** will fail if:

1. Size specified is greater than the size of the previously existing segment. Size specified is less than the system imposed minimum, or greater than the system imposed maximum.

2. No shared memory segment was found matching *key*, and IPC_CREAT was not specified.

3. The kernel was unable to allocate enough memory to satisfy the request.

4. IPC_CREAT and IPC_EXCL were specified, and a shared memory segment corresponding to *key* already exists.

**RETURN VALUES**
Upon successful completion, **shmget**() returns the positive integer identifier of a shared memory segment. Otherwise, -1 is returned

**void *shmat(int shmid, const void *shmaddr, int shmflg);**
Maps a shared memory segment onto your process's address space.

- shmid is the id as returned by shmget() of the shared memory segment you wish to attach.
- Addr is the address where you want to attach the shared memory. For simplicity we will pass NULL. NULL means that kernel itself will decide where to attach it to address space of the process.

**Shmat() will fail if:**
1. No shared memory segment was found corresponding to the given id.

**RETURN VALUES**

Upon success, **shmat**() returns the address where the segment is attached; otherwise, -1 is returned and *errno* is set to indicate the error.

Upon success, **shmdt**() returns 0; otherwise, -1 is returned and *errno* is set to indicate the error.

**int shmdt(void *addr);**

This system call is used to detach a shared memory region from the process's address space.

- Addr is the address of the shared memory

**shmdt will fail if:**

1. The address passed to it does not correspond to a shared region.

**RETURN VALUES**

On success, shmdt() returns 0; on error -1 is returned

**How To Delete Shared Memory Region:**

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

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

For Deletion we will use **IPC_RMID** flag.

**IPC_RMID** marks the segment to be destroyed. The segment will actually be destroyed only after the last process detaches it (The caller must be the owner or creator of the segment, or be privileged). The *buf* argument is ignored.

**Return Value:**
For IPC_RMID operation, 0 is returned on success; else -1 is returned.

```
Shmctl(shmid, IPC_RMID, NULL);
```