Shared memory

- Shared memory allows 2 or more processes to share a memory region or segment of memory for reading and writing purposes.

- The problem with pipes, fifo → message queues is that mode switches are involved as the data has to pass from one process buffer to the kernel buffer → then to another process buffer.

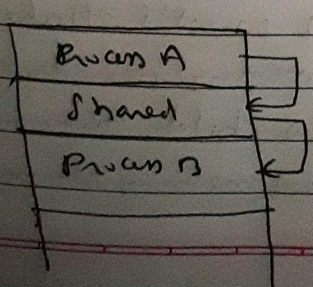
- Since, access to user space memory does not require a mode switch, therefore, shared memory is considered as one of the quickest means of IPC.

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

- Shmget() creates a new shared memory segment or obtains the identifier of an existing segment. The contents of a newly created shared memory segment are initialised to 0. The return value is the ID of the shared memory segment.

- 2nd argument 'size' specifies the desired size of the segment in bytes. Kernel round it up to the next multiple of the system page size. If shared memory is already created then this parameter has no significance.

- Shmflag is for permission



→ void * Shmat (int shmid, const void *shmaddr, int shmflg)

→ Shmattach function system call attaches the shared memory segment identified by shmid to the address space of the calling process.

→ Shmaddr is the address where the memory segment will be attached. If we want the OS kernel to select the suitable address, we keep it NULL.

→ Shmflg is for permission. 0 for read-write access.

→ on success, the func^n returns the address at which the shared memory segment is attached, ~~which is attached~~

~~Shmdt~~ int Shmdt (const void *Shmaddr)
Shmdetach

→ when a process no longer needs to access the shared memory segment, it can call Shmdt() to detach the segment from its address space.

→ Shmaddr - it should be the returned value by a previous call to Shmat()

→ Detaching a shared memory segment is not the same as deleting it. Deletion can be performed using the Shmctl().

→ A child created by fork() inherits its parent's attached shared memory segments. Thus, shared memory provides an easy method of IPC between parent & child. However, after an exec(), all attached shared memory segments are detached.

) Shared memory segments are automatically detached on process termination.

to see the max. no. of shared memory segment possible at a time.

$ cat /Proc/sys/kernel/shmmni

4096

) unlike Pipes, message queues where messages are deleted as the reader reads, but in shared memory messages are ~~still there~~ remain there after reading of the reader. It only get deleted when shared memory is deleted using shmctl().

$ ipcs -m