Shared Memory

- mapping of an area (segment) of memory that will be mapped and shared by more than one process.
- This is by far the fastest form of IPC, because there is no intermediation. Instead, information is mapped directly from a memory segment, and into the addressing space of the calling process.
- A segment can be created by one process, and subsequently written to and read from by any number of processes.

Kernel shm_ds structure

- the kernel maintains a special internal data structure for each shared memory segment which exists within its addressing space (`linux/shm.h`)

```
struct shmid_ds {
     struct ipc_perm shm_perm;        /* operation perms */
     int      shm_segsz;              /* size of segment (bytes) */
     time_t   shm_atime;              /* last attach time */
     time_t   shm_dtime;              /* last detach time */
     time_t   shm_ctime;              /* last change time */
     unsigned short   shm_cpid;       /* pid of creator */
     unsigned short   shm_lpid;       /* pid of last operator */
     short    shm_nattch;             /* no. of current attaches */

     unsigned short    shm_npages;    /* size of segment (pages) */
     unsigned long   *shm_pages;
                      /* array of ptrs to frames -> SHMMAX */
    struct vm_area_struct *attaches; /* descriptors for attaches */
};
```

SYSTEM CALL: shmget()

```
SYSTEM CALL: shmget();
PROTOTYPE: int shmget ( key_t key, int size, int shmflg );
RETURNS: shared memory segment identifier on success
    -1 on error:
   errno = EINVAL (Invalid segment size specified)
           EEXIST (Segment exists, cannot create)
           EIDRM (Segment is marked for deletion, or was removed)
           ENOENT (Segment does not exist)
           EACCES (Permission denied)
           ENOMEM (Not enough memory to create segment)
```

- Primer:

```
int open_segment( key_t keyval, int segsize )
{
      int     shmid;
      if((shmid=shmget(keyval,segsize,IPC_CREAT|0660)) == -1)
        { return(-1); }
      return(shmid);
}
```

SYSTEM CALL: `shmctl()`

- deliver a message to a queue

```
PROTOTYPE: int shmctl ( int shmqid, int cmd, struct shmid_ds *buf );
RETURNS: 0 on success
-1 on error: errno = EACCES (No read permission and cmd is IPC_STAT)
     EFAULT (Address pointed to by buf is invalid with IPC_SET and
             IPC_STAT commands)
    EIDRM  (Segment was removed during retrieval)
    EINVAL (shmqid invalid)
    EPERM  (IPC_SET or IPC_RMID command was issued, but calling
         process does not have write (alter) access to the segment)
```

- Valid command values are:

`IPC_STAT`

Retrieves the `shmid_ds` structure for a segment, and stores it in the address of the `buf` argument

`IPC_SET`

Sets the value of the `ipc_perm` member of the `shmid_ds` structure for a segment. Takes the values from the buf argument.

`IPC_RMID`

Marks a segment for removal. The actual removal itself occurs when the last process currently attached to the segment has properly detached it. If no processes are currently attached to the segment, the removal seems immediate.

SYSTEM CALL: shmat()

- If the addr argument is zero (0), the kernel tries to find an unmapped region. This is the recommended method. An address can be specified, but is typically only used to facilitate proprietary hardware or to resolve conflicts with other apps.

```
PROTOTYPE: int shmat ( int shmid, char *shmaddr, int shmflg);
RETURNS: address at which segment was attached to the process, or
    -1 on error:
errno = EINVAL (Invalid IPC ID value or attach address passed)
        ENOMEM (Not enough memory to attach segment)
        EACCES (Permission denied)
```

- Primer

```
char *attach_segment( int shmid )
{ return(shmat(shmid, 0, 0));
 }
```

SYSTEM CALL: shmdt()

- After a shared memory segment is no longer needed by a process, it should be detached by calling this system call. As mentioned earlier, this is not the same as removing the segment from the kernel! After a detach is successful, the `shm_nattch` member of the associates `shmid_ds` structure is decremented by one. When this value reaches zero (0), the kernel will physically remove the segment.

```
PROTOTYPE: int shmdt ( char *shmaddr );
  RETURNS: -1 on error:
           errno = EINVAL (Invalid attach address passed)
```