Shared Memory (8A)

- Shared Memory

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Young Won Lim
12/29/2012

Shared Memory

- mapping of an area (segment) of memory
- shared by more than one process

- 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
- written to and read from by any number of processes

- the fastest form of IPC (no intermediation)

Kernel shmid_ds Structure

```
/* One shmid data structure for each shared memory segment in the system. */
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 */

        /* the following are private */
        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 */
};
```

Shared Memory System Calls

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

RETURNS: shared memory segment identifier on success

int **shmat** (int shmid, char *shmaddr, int shmflg);

RETURNS: address at which segment was attached to the process, or -1 on error

int **shmdt** (char *shmaddr);

RETURNS: 0 on success, -1 on error

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

RETURNS: 0 on success, -1 on error

shmflg

IPC_CREAT Create the segment if it doesn't already exist in the kernel.

IPC_EXCL When used with IPC_CREAT, fail if segment already exists.

SHM_RND round

SHM_RDONLY readonly.

cmd

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. .

shmget()

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

RETURNS: shared memory segment identifier on success

shmflg

IPC_CREAT Create the segment if it doesn't already exist in the kernel.

IPC_EXCL When used with IPC_CREAT, fail if segment already exists.

shmid = shmget(keyval, segsize, IPC_CREAT | 0660))

shmid = shmget(keyval, segsize, IPC_CREAT | IPC_EXCL | 0660))

shmat()

int **shmat** (int shmid, char *shmaddr, int shmflg);

RETURNS: address at which segment was attached to the process, or -1 on error

shmaddr

If zero (0), the kernel tries to find an unmapped region.

An address can be specified, to facilitate proprietary hardware or to resolve conflicts with other apps.

shmflg

SHM_RND forces a passed address to be page aligned (rounds down to the nearest page size).

SHM_RDONLY the shared memory segment will be mapped in, but marked as readonly.

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

Reading / Writing to the segment → Referencing / Dereferencing the pointer (address)

shmdt()

```
int shmdt ( char *shmaddr );
        RETURNS: 0 on success, -1 on error
```

```
struct shmid_ds {
        struct ipc_perm         shm_perm;
        int                     shm_segsz;
        time_t                  shm_atime;
        time_t                  shm_dtime;
        time_t                  shm_ctime;
        unsigned short          shm_cpid;
        unsigned short          shm_lpid;
        short                   shm_nattch;

        /* the following are private */
        unsigned short          shm_npages;
        unsigned long   *       shm_pages;
        struct vm_area_struct * attaches;
};
```

shm_nattch member is decremented by one.

If it is zero (**0**), then the kernel will physically remove the segment.

not the same as removing the segment from the kernel

shmctl()

```
int shmctl ( int shmqid, int cmd, struct shmid_ds *buf );
        RETURNS: 0 on success, -1 on error
```

```
int cmd;
int shmid;
struct shmid_ds my_ds;

shmid = ...
cmd = ...
if ((rtrn = shmctl(shmid, cmd, shmid_ds)) == -1) {
    perror("shmctl: shmctl failed");
    exit(1);
   }


shmctl(shmid, IPC_STAT, &my_ds);  // read

my_ds.shm_perm.uid = new_uid;
my_ds.shm_perm.gid = new_gid;
shmctl(shmid, IPC_SET, &my_ds);  // write

shmctl(shmid, IPC_RMID, 0);  // remove
```

cmd

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. .

```
struct shmid_ds {
   struct ipc_perm      shm_perm;
      ...
};

struct ipc_perm
{
  key_t       key;
  ushort      uid;      /* owner euid and egid */
  ushort      gid;
  ushort      cuid;     /* creator euid and egid */
  ushort      cgid;
  ushort      mode;     /* access modes */
  ushort      seq;      /* slot usage seq number */
};
```

Reference

References

[1] http://en.wikipedia.org/
[2] http://www.tldp.org/LDP/lpg/node46.html