UNIX SYSTEM PROGRAMMING

IPC - Shared MEMORY

- Shared memory allows multiple processes to share virtual memory space
- This is the fastest way for processes to communicate with one another.
- one process creates or allocates the shared memory segment.
- The size and access permissions for the segment are set when it is created.
- The process then attaches the shared segment, causing it to be mapped into its current data space.

Creating a Shared Memory Segment

shmget() is used to obtain access to a shared memory segment

2	N	Manual Section		<sys ipc.h=""> <sys shm.h=""></sys></sys>	Include File(s)
int shmget(key_t key, int size,int shmflg); Summary					
Return		Failure		Success	
			-1	Shared memory identifier.	

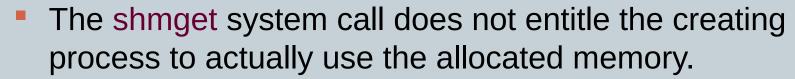
Summary of the shmget System Call.

Creating a Shared Memory Segment

```
#include <sys/shm.h>
#include <sys/ipc.h>
int main()
   key t \text{ key} = 300;
   system("ipcs -m");
   int shmid = shmget(key , 50 , IPC_CREAT | 0666);
   if(shmid == -1)
               perror(" ");
       exit(1);
   system("ipcs -m");
```

To view the list of shared memory segment, use **ipcs -m**

- The shmget system call creates a new shared memory segment if
 - The value for its first argument, key, is the symbolic constant IPC_PRIVATE, or
 - the value key is not associated with an existing shared memory identifier and the IPC_CREAT flag is set as part of the shmflg argument
- The argument size determines the size in bytes of the shared memory segment.
- If we are using shmget to access an existing shared memory segment, size can be set to 0, as the segment size is set by the creating process



- It merely reserves the requested memory.
- To be used by the process, the allocated memory must be attached to the process using a separate system call.

Attaching and Detaching a Shared Memory Segment

• shmat, is used to attach (map) the referenced shared memory segment into the calling process's data segment.

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

- shmid, is a valid shared memory identifier achieved from shmget.
- Shmaddr
 - If a nonzero value is given, shmat uses this as the attachment address for the shared memory segment.
 - If shmaddr is 0(NULL), the system picks the attachment address.
 - In most situations, it is advisable to use a value of 0 and have the system pick the address.
- Shmflg: 0: Read Write, SHM_RDONLY: Read only
- shmdt, is used to detach the calling process's data segment from the shared memory segment

int shmdt(const void *shmaddr);

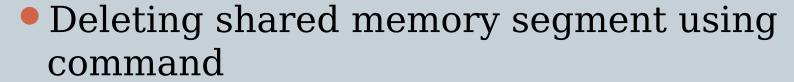
```
int main()
    key t \text{ key} = 301;
    int shmid = shmget(IPC\_PRIVATE , 50 , IPC\_CREAT | 0600);
   int *ptr = shmat(shmid,NULL,0);
   system("ipcs");
     if((int)ptr != -1)
          int i;
          for(i = 0; i < 5; i++)
               ptr[i] = i + 5;
               //ptr++;
          shmdt(ptr);
```

Shared Memory Control

- The shmctl system call permits the user to perform a number of generalized control operations on an existing shared memory segment and on the system shared memory data structure.
- •int shmctl(int shmid, int cmd, struct shmid_ds *buf);
- shmid, is a valid shared memory segment identifier generated by a prior shmget system call
- cmd, specifies the operation shmctl is to perform
- buf, is a reference to a structure of the type shmid ds
- Returns 0 on success; otherwise, it returns -1

IPC_RMID : Destroy a segment.
IPC_STAT : copy the information into the buffer.
 struct shmid_ds buf;
 shmctl(shmid,IPC_RMID,&buf);

```
int main()
key t key = 0;
printf("Enter the key : ");
scanf(" %d",&key);
int shmid = shmget(key, 50, IPC CREAT | 0666);
struct shmid ds buf;
shmctl(shmid,IPC STAT,&buf);
printf("Size = %d \n",buf.shm segsz);
printf("Last attach time = %s \n",ctime(\&buf.shm atime));
printf("Last detach time = %s \n",ctime(\&buf.shm dtime));
printf("pid of creator = %d \n",buf.shm cpid);
```



• ipcrm -M key