Operating system
Lab 4
Shared memory and Semaphores

Shared memory

You already know what a shared memory is. We had a bit of problem last time when we wanted to send some characters from the child to the parent or vv. We used pipe to solve the problem. Here are a few other ways to do the same thing. The first one uses the shared memory. We have used the built in function shmget to get the shared memory. We have used the shared memory to pass data from the child to the parent.

```
#include <stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>

const key_t SHMKEY=7890;
const int PERMS =0666;
char exflag=0;
void main()
{
        int shmid;
        char* data;
        shmid=shmget(SHMKEY,1,PERMS|IPC_CREAT);
        if(shmid<0)
                printf("can't get the shared memory");

        data=(char*)shmat(shmid,0,0);
        *data=0;
        if(!fork())
        {
                printf("press any key to exit…..\n");
                scanf("%c",&data,1);
                printf("child exiting….\n");
                exit(0);
        }
else{
        while(!data);
        printf("Received %c from child\n", *data);
        shmdt(data);
        shmctl(shmid,IPC_RMID,0);
        printf("Parent exititn….");
        exit(0);
        }
}
```

Semaphore

As in the earlier case, rather than defining the term I'll jump in to the task. Type ipcs –s at the command prompt. Now run the following program.

```
#include<sys/types.h>
#include<sys/ipc.h>

main()
{
        int semid,key,nsem;
key=(key_t)0x20;
nsem=0;
semid=semget(key,nsem,ipc_creat|0666);
printf("created semaphore with id: %d",semid);
}
```

Change the parameter nsem to 1. and run the above program. Check the semaphore status in the command prompt after each prompt. Comment on the result.

```
#include<sys/types.h>
#include<sys/ipc.h>

main()
{
        int semid,key,nsem,flag;

        nsem=1;
        flag=0666|IPC_CREAT;
        for(i=0;;i++)
        {
        key=(key_t)I;
        semid=semget(key,nsem,flag);

        if(semid>0)
                printf("Created semaphore with ID:%d", semid);
        else
        {
        printf("Maximum number of semaphore set are %d\n",i);
        exit(0);
        }
}
```

What is the maximum number of semaphore set that can be created?

Now lets see how data can be transferred from one process to another with a wait on a semaphore. We use the build-in function semop() to have a wait on the semaphore.

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/sem.h>

Const key_t SEMKEY=4567;
Const int perms=0666;

Char exflag=0;

Void main()
{
        int semid;
semid=semget(SEMKEY,1,PERMS|IPC_CREAT);

if(semid<0)
                {printf(can’t create semaphore\n”);exit(1);}

if(!fork())
{
                char data;
                sembuf* sbuf=(sembuf*)malloc(sizeof(sembuf));

sbuf->sem_num=0;
sbuf->sem_op=1;
sbuf->sem_flg=0;
printf(“press any key to exit….\n”);
scanf(“%c”,&data);
semop(semid,sbuf,1);

printf(“child exiting”);

exit(0);
}
else
{
                sembuf* sbuf= (sembuf*)malloc(sizeof(sembuf));

                sbuf->sem_num=0;
                sbuf->sem_op=-1;
                sbuf->sem_flg=0;

        semop(semid,sbuf,1);
```

```
                printf(“child released semaphore\n”);
                printf(“parent exitint”);
                exit(0);
        }

        }
        }
```

Do you know any other methods to do the same thing? If you do then mention here, if possible, with an example implementation.