Department of Computer Science and Engineering
SSN College of Engineering

141451 – Operating Systems Lab - Manual

Exercise : InterProcess Communications using Shared Memory, Pipe and Message Queues.

Study the following system calls.

Shared memory - shmget, shmat, shmdt, shmctl
Pipe - pipe, mkfifo, mknod, open, read, write, close
Message Queue – msgget, msgsnd, msgrcv, msgctl

1. Develop an application for getting a name in parent and convert it into uppercase in child using shared memory.
2. Develop an echo client / server application using shared memory.
3. Develop an application to get a number in child and find the reverse of that number in parent using pipe.
4. Develop a client / server chat application using pipes
5. Develop an application to get a sentence in parent and find the no. of words in child using message queues.
6. Develop a client / server application to get 'n' inputs in server and write them into a file in client using message queue. Server should wait until client gives “over” message.

Some Examples :
Shared memory :

```
#include <sys/ipc.h>
# define NULL 0
#include <sys/shm.h>
#include <sys/types.h>
# include<unistd.h>
# include<stdio.h>
# include<stdlib.h>
# include<string.h>
#include <sys/wait.h>
#include <stdio_ext.h>

 // parent writing a char in shared memory and child reads it and prints it.

int main()
{
```

```
int pid;
char *a,*b,c;
  int id,n,i;
// you can create a shared memory between parent and child here or you can create inside them separately.

 id=shmget(111,50,IPC_CREAT | 00666);

 pid=fork();
 if(pid>0)
{
// id=shmget(111,50,IPC_CREAT | 00666);

 a=shmat(id,NULL,0);
 a[0]='d';
 wait(NULL);
 shmdt(a);
}
else
{
sleep(3);
//id=shmget(111,50,0);

b=shmat(id,NULL,0);
printf("\n child  %c\n",b[0]);
shmdt(b);
}

}
```

Pipe

```
#include<stdio.h>
#include <sys/types.h>
#include <stdio_ext.h>
#include<unistd.h>

// child writing input in pipe and parent reading it.
main()
{
 int pid;
char n[10],n1[10];
int fd[2];
pipe(fd);
pid=fork();

if(pid==0)
```

```
 {
  printf("child enter input\n");
     scanf("%s",n);
     close(fd[0]);
     write(fd[1],&n,strlen(n));
     exit(0);
}
else
{

 sleep(3);
 close(fd[1]);
 read(fd[0],&n1,sizeof(n1));
 printf("\n parent reading %s \n",n1);
}

}
```

Named Pipe:

```
#include<stdio.h>
#include <sys/types.h>
#include <stdio_ext.h>
#include <sys/stat.h>
#include <fcntl.h>
#include<unistd.h>

// child writing input and parent reading it.
main()
{
 int pid;
char n[10],n1[10];
int rd,wd;
mkfifo("mynamedpipe",00666);
pid=fork();
if(pid==0)
 {
  printf("child enter input\n");
   scanf("%s",n);
  wd=open("mynamedpipe",O_WRONLY);
  write(wd,&n,strlen(n));
  close(wd);
  exit(0);
}
else
{
 sleep(3);
 rd=open("mynamedpipe",O_RDONLY);
 read(rd,&n1,sizeof(n1));
```

```
 printf("\n parent reading %s \n",n1);
 close(rd);
}
}
```

Message Queue

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

 #include <sys/ipc.h>

#include <sys/msg.h>

#include <unistd.h>

# include <stdio.h>

# include<string.h>


struct msg

{

 long type;

 char data[10];

}buf,buf1;


// CHILD WRITING INPUT MSG AND PARENT READING IT
main()

{

 int id,pid;

   id=msgget(111,IPC_CREAT | 00666);



   pid=fork();
```

```
if (pid==0)
 {
printf("\n Enter a  name \n");
scanf("%s",buf.data);
buf.type=1;
msgsnd(id,&buf,strlen(buf.data),IPC_NOWAIT);

 }

else
{
wait(0);
msgrcv(id,&buf1,sizeof(buf1),1,IPC_NOWAIT);
printf("\n PARENT %s\n",buf1.data);

}

}
```