**P1 (Send Data from one process to another)**

#include<stdio.h>  
#include<stdlib.h>  
#include<unistd.h>  
#include<string.h>

int main()

{

FILE \*rd;

char buffer[50];

sprintf(buffer,"name first");

rd=popen("wc -c","w"); // wc -c -> is the process which counts the number of characters passed. 2nd parameter is "w" which means pipe is opened in writing mode

fwrite(buffer,sizeof(char),strlen(buffer),rd); // to write the data into the pipe

pclose(rd);

}unistd.h>

#include

**P2 (Receive data from one process)**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<string.h>

int main()

{

FILE \*rd;

char buffer[50];

rd=popen("ls","r"); //pipe opened in reading mode

fread(buffer, 1, 40, rd);man //will read only 50 characters

printf("%s\n", buffer);

pclose(rd);

}

**P3 (Send data from parent to child)**

#include<stdio.h>  
#include<unistd.h>  
#include<sys/types.h>  
int main()  
{  
int fd[2],n;  
char buffer[100];  
pid\_t p;  
pipe(fd); //creates a unidirectional pipe with two end fd[0] and fd[1]  
p=fork();  
if(p>0) //parent  
{  
printf("Parent Passing value to child\n");  
write(fd[1],"hello\n",6); //fd[1] is the write end of the pipe  
}  
else // child  
{  
printf("Child printing received value\n");  
n=read(fd[0],buffer,100); //fd[0] is the read end of the pipe  
write(1,buffer,n);  
}  
}

**P4 (Message Passing from shared memory)**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/shm.h>

#include<string.h>

int main()

{

int i;

void \*shared\_memory;

char buff[100];

int shmid;

shmid=shmget((key\_t)2345, 1024, 0666|IPC\_CREAT); //creates shared memory segment with key 2345, having size 1024 bytes. IPC\_CREAT is used to create the shared segment if it does not exist. 0666 are the permisions on the shared segment

printf("Key of shared memory is %d\n",shmid);

shared\_memory=shmat(shmid,NULL,0); //process attached to shared memory segment

printf("Process attached at %p\n",shared\_memory); //this prints the address where the segment is attached with this process

printf("Enter some data to write to shared memory\n");

read(0,buff,100); //get some input from user

strcpy(shared\_memory,buff); //data written to shared memory

printf("You wrote : %s\n",(char \*)shared\_memory);

}

**P5.1 (Message Queues)**

#include<stdlib.h>

#include<stdio.h>

#include<string.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/msg.h>

#define MAX\_TEXT 512 //maximum length of the message that can be sent allowed

struct my\_msg{

long int msg\_type;

char some\_text[MAX\_TEXT];

};

int main()

{

int running=1;

int msgid;

struct my\_msg some\_data;

char buffer[50]; //array to store user input

msgid=msgget((key\_t)14534,0666|IPC\_CREAT);

if (msgid == -1) // -1 means the message queue is not created

{

printf("Error in creating queue\n");

exit(0);

}

while(running)

{

printf("Enter some text:\n");

fgets(buffer,50,stdin);

some\_data.msg\_type=1;

strcpy(some\_data.some\_text,buffer);

if(msgsnd(msgid,(void \*)&some\_data, MAX\_TEXT,0)==-1) // msgsnd returns -1 if the message is not sent

{

printf("Msg not sent\n");

}

if(strncmp(buffer,"end",3)==0)

{

running=0;

}

}

}

**P 5.2**

#include<stdlib.h>

#include<stdio.h>

#include<string.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/msg.h>

struct my\_msg{

long int msg\_type;

char some\_text[BUFSIZ];

};

int main()

{

int running=1;

int msgid;

struct my\_msg some\_data;

long int msg\_to\_rec=0;

msgid=msgget((key\_t)14534,0666|IPC\_CREAT);

while(running)

{

msgrcv(msgid,(void \*)&some\_data,BUFSIZ,msg\_to\_rec,0);

printf("Data received: %s\n",some\_data.some\_text);

if(strncmp(some\_data.some\_text,"end",3)==0)

{

running=0;

}

}

msgctl(msgid,IPC\_RMID,0);

}

**P 6.1 (Named Pipes)**

#include<stdio.h>

#include<sys/types.h>

#include<sys/stat.h>

int main()

{

int res;

res = mkfifo("fifo1",0777); //creates a named pipe with the namefifo1

printf("named pipe created\n");

}

**P 6.2**

#include<unistd.h>

#include<stdio.h>

#include<fcntl.h>

int main()

{

int res,n;

res=open("fifo1",O\_WRONLY);

write(res,"Message",7);

printf("Sender Process %d sent the data\n",getpid());

}

**P 6.3**

#include<unistd.h>

#include<stdio.h>

#include<fcntl.h>

int main()

{

int res,n;

char buffer[100];

res=open("fifo1",O\_RDONLY);

n=read(res,buffer,100);

printf("Reader process %d started\n",getpid());

printf("Data received by receiver %d is: %s\n",getpid(), buffer);

}

The C library function **int sprintf(char \*str, const char \*format, ...)** sends formatted output to a string pointed to, by **str**.

Declaration

Following is the declaration for sprintf() function.

int sprintf(char \*str, const char \*format, ...)