**Week – 3**

**1.Write a C program to demonstrate named pipe**

**Code:**

**1st Process:**

#include<stdio.h>

#include<string.h>

#include<fcntl.h> // file control

#include<sys/stat.h> //system status

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

#include<unistd.h>

int main(){

char \*myfifo = "/tmp/myfifo3262";

mkfifo(myfifo,0666);

int fd = open(myfifo,O\_WRONLY);

char arr2[80];

printf("process1:");

fgets(arr2,80,stdin);

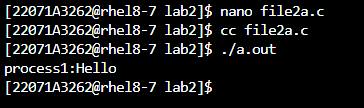
write(fd,arr2,strlen(arr2)+1);

close(fd);

return 0;

}

**Output:**



**2nd Process:**

#include<stdio.h>

#include<string.h>

#include<fcntl.h>

#include<sys/stat.h>

#include<sys/types.h>

#include<unistd.h>

int main(){

char \*myfifo = "/tmp/myfifo3262";

mkfifo(myfifo,0666);

int fd = open(myfifo,O\_RDONLY);

char arr1[80];

read(fd,arr1,sizeof(arr1));

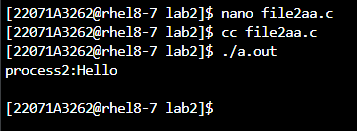
printf("process2:%s\n",arr1);

close(fd);

return 0;

}

**Output:**



**2. Write a C program to demonstrate unnamed pipe**

**Code:**

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<string.h>

#include<unistd.h>

int main(){

pid\_t pid;

pid = fork();

int fd[2];

pipe(fd);

if(pid == 0){

close(fd[0]);

dup2(fd[1],STDOUT\_FILENO);

system("ls -l");

close(fd[1]);

}

if(pid>0){

close(fd[1]);

dup2(fd[0],STDIN\_FILENO);

system("sort -r");

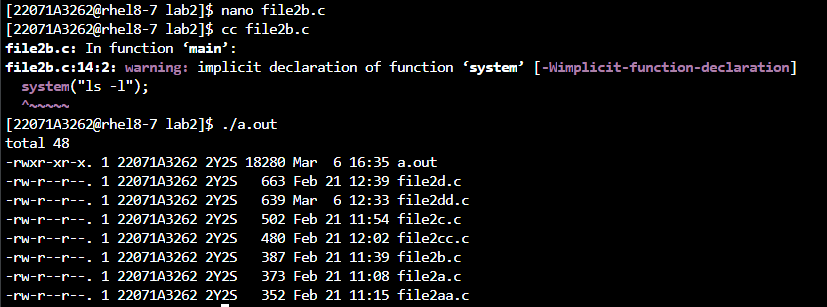
close(fd[0]);

}

return 0;

}

**Output:**



**3. Write a C program to demonstrate shared memory**

**Code:**

**Process1:**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<string.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/shm.h>

#define SHMSZ 27

int main(){

int shmid;

char \*shm,\*s;

if((shmid = shmget((key\_t)5577,SHMSZ,IPC\_CREAT|0666)) < 0){

perror("shmat");

exit(0);

}

if((shm = shmat(shmid,NULL,0)) == (char\*)-1){

perror("shmat");

exit(0);

}

s = shm;

for(char c = 'a';c <= 'z';c++){

\*s++ = c;

}

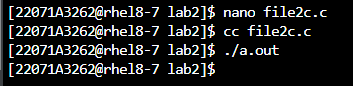
\*s = '\0';

exit(1);

return 0;

}

**Output:**



**Process 2:**

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/shm.h>

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<string.h>

#define SHMSZ 27

int main(){

int shmid;

char \*shm,\*s;

if((shmid = shmget((key\_t)5577,SHMSZ,0666)) < 0){

perror("shmget");

}

if((shm = shmat(shmid,NULL,0)) == (char\*)-1){

perror("shmat");

}

for(s = shm;\*s != '\0';s++){

putchar(\*s);

}

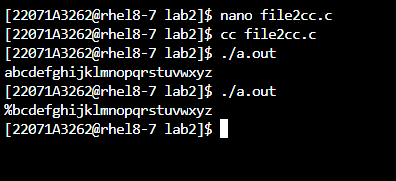
putchar('\n');

\*shm = '%';

return 0;

}

**Output:**



**4. Write a C program to demonstrate message queue**

**Code:**

**Message\_queue\_sender:**

#include<stdio.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/msg.h>

#include<string.h>

struct msgbuf {

long mtype;

char mtext[10];

};

int main() {

struct msgbuf msgsend = {0, '\0'};

int msqid = msgget((key\_t)3262, IPC\_CREAT|0666);

if(msqid == - 1)

perror("msgget");

for(int i=1;i<=3;i++) {

printf("Enter message %d's type\n", i);

scanf("%d", &msgsend.mtype);

printf("Enter message %d\n", i);

scanf("%s", msgsend.mtext);

int len = strlen(msgsend.mtext);

int ret = msgsnd(msqid,&msgsend,len,0);

if(ret == -1)

perror("msgsnd");

else

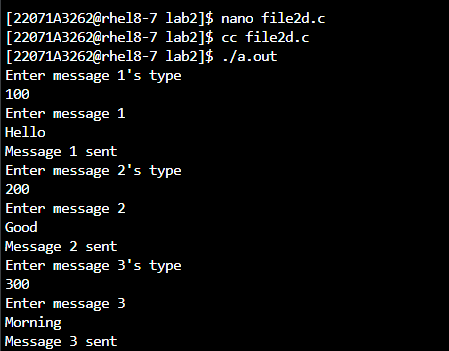
printf("Message %d sent\n", i);

}

return 0;

}

**Output:**



**Message\_queue\_receiver:**

#include<stdio.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/msg.h>

#include<string.h>

struct msgbuf {

long mtype;

char mtext[10];

};

int main() {

int msqid, type;

struct msgbuf msgread = {0, '\0'};

msqid = msgget((key\_t)3258, IPC\_CREAT|0666);

if(msqid == - 1)

perror("msgget");

for(int i=1;i<=3;i++) {

printf("\nEnter message %d's type\n", i);

scanf("%d", &type);

int ret = msgrcv(msqid, &msgread, sizeof(msgread.mtext), type, 0);

if(ret == -1)

perror("msgrcv");

else

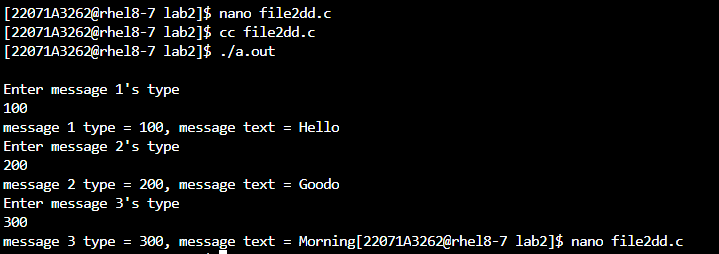
printf("message %d type = %d, message text = %s", i, msgread.mtype, msgread.mtext);

}

return 0;

}

**Output:**



**5.Write a C program to demonstrate the use of unnamed pipe for the following scenario:**

**Child process should read a string from the std input and write the string to the parent process. In parent process read the string from child process and check the string is palindrome or not**

**Code:**

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <string.h>

int isPalindrome(char \*str) {

int len = strlen(str);

for (int i = 0; i < len / 2; i++) {

if (str[i] != str[len - i - 1]) {

return 0;

}

}

return 1;

}

int main() {

pid\_t pid;

int fd[2], c;

pipe(fd);

char inputString[100];

pid = fork();

if (pid == 0) {

close(fd[0]);

printf("Child Process: Enter a string: ");

fgets(inputString, sizeof(inputString), stdin);

write(fd[1], inputString, strlen(inputString) + 1);

close(fd[1]);

} else if (pid > 0) {

close(fd[1]);

read(fd[0], inputString, sizeof(inputString));

close(fd[0]);

printf("Parent Process: Received string from child process: %s", inputString);

if (isPalindrome(inputString)) {

printf("Parent Process: The string is a palindrome.\n");

} else {

printf("Parent Process: The string is not a palindrome.\n");

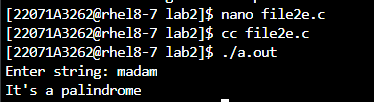
}

}

return 0;

}

**Output:**

****

**6. Write a C program to demonstrate the use of named pipe for the following scenario:**

**1st process should read a string from the std input and write the string to the 2nd process. In 2nd process read the string from the 1st process and check the string is palindrome or not**

**Code:**

**1st Process:**

#include <stdio.h>

#include <string.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

int main() {

int fd;

char \*myfifo = "/tmp/myfifo3262";

char arr2[80];

mkfifo(myfifo, 0666);

fd = open(myfifo, O\_WRONLY);

printf("Process 1: Enter a string: ");

fgets(arr2, 80, stdin);

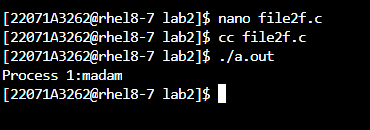
write(fd, arr2, strlen(arr2) + 1);

close(fd);

return 0;

}

**Output:**

****

**2nd Process:**

#include <stdio.h>

#include <string.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

int isPalindrome(char \*str) {

int len = strlen(str);

for (int i = 0; i < len / 2; i++) {

if (str[i] != str[len - i - 1]) {

return 0;

}

}

return 1;

}

int main() {

int fd;

char \*myfifo = "/tmp/myfifo3262";

char arr1[80];

mkfifo(myfifo, 0666);

fd = open(myfifo, O\_RDONLY);

read(fd, arr1, sizeof(arr1));

close(fd);

printf("Process 2: Received string from Process 1: %s\n", arr1);

if (isPalindrome(arr1)) {

printf("Process 2: The string is a palindrome.\n");

} else {

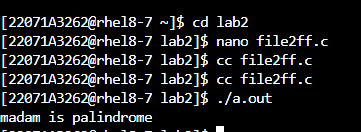
printf("Process 2: The string is not a palindrome.\n");

}

return 0;

}

**Output:**

****

**7.Write a C program to write a String in a shared memory**

**Code:**

**Writer process:**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<string.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/shm.h>

#define SHMSZ 100

int main() {

int shmid;

key\_t key;

char \*shm, \*s;

key = 5577;

if ((shmid = shmget(key, SHMSZ, IPC\_CREAT | 0666)) < 0) {

perror("shmget");

exit(1);

}

if ((shm = shmat(shmid, NULL, 0)) == (char\*)-1) {

perror("shmat");

exit(1);

}

printf("Writer Process: Enter a string: ");

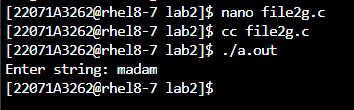
fgets(shm, SHMSZ, stdin);

shmdt(shm);

return 0;

}

**Output:**

****

**Reader process:**

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/shm.h>

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<string.h>

#define SHMSZ 100

int main() {

int shmid;

key\_t key;

char \*shm, \*s;

key = 5577;

if ((shmid = shmget(key, SHMSZ, 0666)) < 0) {

perror("shmget");

exit(1);

}

if ((shm = shmat(shmid, NULL, 0)) == (char\*)-1) {

perror("shmat");

exit(1);

}

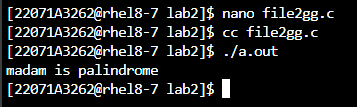
printf("Reader Process: Data read from shared memory: %s\n", shm);

shmdt(shm);

return 0;

}

**Output:**

****

**8.Write a C program to read the string which is written in above question (6) and check whether it is a palindrome or not**

**Code:**

#include <stdio.h>

#include <string.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#define SHMSZ 100

int isPalindrome(char \*str) {

int len = strlen(str);

for (int i = 0; i < len / 2; i++) {

if (str[i] != str[len - i - 1]) {

return 0;

}

}

return 1;

}

int main() {

int shmid;

key\_t key;

char \*shm;

key = 5577;

if ((shmid = shmget(key, SHMSZ, 0666)) < 0) {

perror("shmget");

return 1;

}

if ((shm = shmat(shmid, NULL, 0)) == (char\*)-1) {

perror("shmat");

return 1;

}

printf("Reader Process: Data read from shared memory: %s\n", shm);

if (isPalindrome(shm)) {

printf("Reader Process: The string is a palindrome.\n");

} else {

printf("Reader Process: The string is not a palindrome.\n");

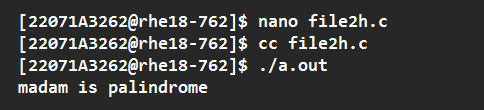
}

shmdt(shm);

return 0;

}

**Output:**

****

**9.Write a C program to demonstrate the use of message queues in four processes (1 act as broadcaster and other 3 will receive messages)**

**Code:**

**Broadcaster:**

#include<stdio.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/msg.h>

#include<string.h>

struct msgbuf {

 long mtype;

 char mtext[10];

};

int main() {

int j, msqid, len, ret, i;

struct msgbuf msgsend = {0, '\0'};

for(j=1000;j<=3000;j+=1000) {

msqid = msgget((key\_t)j, IPC\_CREAT|0666);

if(msqid == - 1)

 perror("msgget");

for(i=1;i<=2;i++) {

  printf("Enter message %d's type for queue %d: ", i, msqid);

  scanf("%ld", &msgsend.mtype);

  printf("Enter message %d: ", i);

  scanf("%s", msgsend.mtext);

 len = strlen(msgsend.mtext);

  ret = msgsnd(msqid,&msgsend,len,0);

  if(ret == -1)

   perror("msgsnd");

  else

   printf("Message %d sent\n", i);

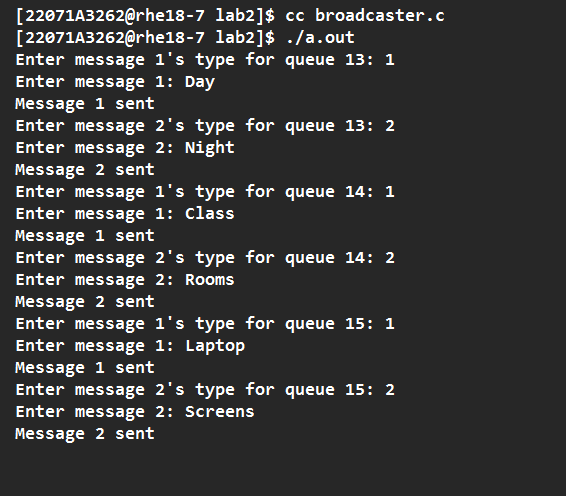
}

}

return 0;

}

**Output:**

****

**Receiver:**

#include<stdio.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/msg.h>

#include<string.h>

struct msgbuf {

  long mtype;

char mtext[10];

};

int main() {

int msqid, len, ret, i, type, j,m;

struct msgbuf msgread = {0, '\0'};

printf("Enter message queue id: ");

scanf("%d", &m);

msqid = msgget((key\_t)m, IPC\_CREAT|0666);

if(msqid == - 1)

  perror("msgget");

for (i=1;i<=2;i++) {

  printf("Enter message %d's type: ", i);

  scanf("%d", &type);

  ret = msgrcv(msqid, &msgread, sizeof(msgread.mtext), type, 0);

  if(ret == -1)

   perror("msgrcv");

  else

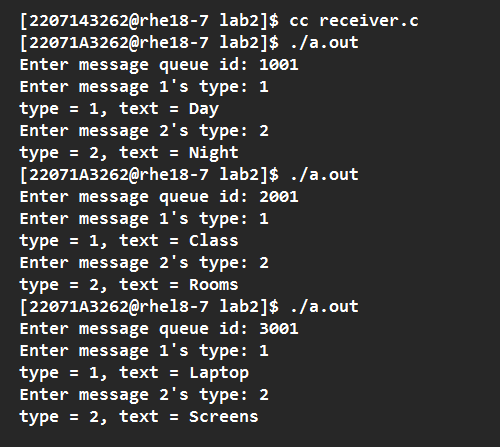
   printf("type = %ld, text = %s\n", msgread.mtype, msgread.mtext);

}

return 0;

}

**Output:**

****