Bài 1:

**Share memory:**

#include <stdio.h>

#include <unistd.h>

#include <limits.h>

#include <string.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#define SIZE 256

int main(int argc, char\* argv[])

{

int \*shm, shmid, k,pid;

key\_t key;

if((key=ftok(".",65))==-1){

perror("Key created.\n");

return 1;

}

shmid = shmget(key, SIZE, IPC\_CREAT | 0666);

if (shmid == -1) {

perror("Shared memory created.\n");

return 2;

}

shm = (int\*) shmat(shmid, 0, 0);

pid = fork();

if(pid==0) { // child

shm[0] = atoi(argv[1]);

sleep(4);

printf("%d!= %d\n", shm[0],shm[1]);

shmdt((void\*) shm);

shmctl(shmid, IPC\_RMID, (struct shmid\_ds\*) 0);

return 0;

}

else if(pid >0) { // parent

sleep(2);

int i,cnt=1;

for(i=1;i<=shm[0];i++){

cnt\*=i;

}

shm[1]=cnt;

shmdt((void\*) shm);

sleep(5);

return 0;

}

else { perror("Fork failed."); return 4; }

return 0;

}

Text

Description automatically generated

**Message queue:**

**File: Writer.c**

//Write

// C Program for Message Queue (Writer Process)

#include <stdio.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <string.h>

// structure for message queue

struct mesg\_buffer {

long mesg\_type;

char mesg\_text[100];

} message;

int main()

{

key\_t key;

int msgid;

// ftok to generate unique key

key = ftok("msg.txt",1);

// msgget creates a message queue

// and returns identifier

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

message.mesg\_type = 1;

printf("Write Data : ");

fgets(message.mesg\_text, sizeof(message.mesg\_text), stdin);

// msgsnd to send message

msgsnd(msgid, &message, sizeof(message),0);

return 0;

}

**File reader.c**

//read

// C Program for Message Queue (Reader Process)

#include <stdio.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <string.h>

// structure for message queue

struct mesg\_buffer {

long mesg\_type;

char mesg\_text[100];

} message;

int main()

{

key\_t key;

int msgid;

// ftok to generate unique key

key = ftok("msg.txt",1);

// msgget creates a message queue

// and returns identifier

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

// msgrcv to receive message

msgrcv(msgid, &message, sizeof(message),1,0);

int cnt=1,i;

int n=atoi(message.mesg\_text);

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

cnt\*=i;

}

printf("%d!=%d",n,cnt);

// to destroy the message queue

msgctl(msgid, IPC\_RMID, NULL);

return 0;

}

A screenshot of a computer

Description automatically generated

Bài 2:

**Share memory:**

#include <stdio.h>

#include <unistd.h>

#include <limits.h>

#include <string.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#define SIZE 256

int main(int argc, char \*argv[])

{

int \*shm, shmid, k, pid;

key\_t key;

if ((key = ftok(".", 65)) == -1)

{

perror("Key created.\n");

return 1;

}

shmid = shmget(key, SIZE, IPC\_CREAT | 0666);

if (shmid == -1)

{

perror("Shared memory created.\n");

return 2;

}

shm = (int \*)shmat(shmid, 0, 0);

pid = fork();

if (pid == 0)

{ // child

shm[0] = atoi(argv[1]);

shm[1] = atoi(argv[2]);

shm[2] = (int)(argv[3][0]);

sleep(3);

switch (shm[2])

{

case 43:

printf("%d+%d=%d\n", shm[0],shm[1],shm[3]);

break;

case 45:

printf("%d-%d=%d\n", shm[0],shm[1],shm[3]);

break;

case 120:

printf("%d\*%d=%d\n", shm[0],shm[1],shm[3]);

break;

case 47:

printf("%d/%d=%d\n", shm[0],shm[1],shm[3]);

break;

}

shmdt((void \*)shm);

shmctl(shmid, IPC\_RMID, (struct shmid\_ds \*)0);

return 0;

}

else if (pid > 0)

{ // parent

printf("Data %d",shm[2]);

sleep(1);

if(shm[2]==43){

shm[3]=shm[1]+shm[0];

}else if(shm[2]==45){

shm[3]=shm[1]-shm[0];

}else if(shm[2]==120){

shm[3]=shm[1]\*shm[0];

}else if(shm[2]==47){

shm[3]=shm[0]\*1.0/shm[1];

}

shmdt((void \*)shm);

sleep(5);

return 0;

}

else

{

perror("Fork failed.");

return 4;

}

return 0;

}

Text

Description automatically generated

**Message queue:**

**File: Writer.c**

**// C Program for Message Queue (Writer Process)**

**#include <stdio.h>**

**#include <sys/ipc.h>**

**#include <sys/msg.h>**

**#include <string.h>**

**// structure for message queue**

**struct mesg\_buffer**

**{**

**long mesg\_type;**

**char mesg\_text[100];**

**} message;**

**int main()**

**{**

**key\_t key;**

**int msgid;**

**// ftok to generate unique key**

**key = ftok("msg.txt", 1);**

**// msgget creates a message queue**

**// and returns identifier**

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

**message.mesg\_type = 1;**

**printf("Write Data :");**

**fflush(stdin);**

**fgets(message.mesg\_text, sizeof(message.mesg\_text), stdin);**

**msgsnd(msgid, &message, sizeof(message), 0);**

**// display the message**

**printf("Data send is : %s \n", message.mesg\_text);**

**return 0;**

**}**

**File reader.c**

// C Program for Message Queue (Reader Process)

#include <stdio.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <string.h>

// structure for message queue

struct mesg\_buffer

{

long mesg\_type;

char mesg\_text[100];

} message;

int main()

{

key\_t key;

int msgid;

// ftok to generate unique key

key = ftok("msg.txt", 1);

// msgget creates a message queue

// and returns identifier

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

// msgrcv to receive message

msgrcv(msgid, &message, sizeof(message), 1, 0);

// message.mesg\_text

int tmp[20];

int i, cnt = 0;

for (i = 0; i < strlen(message.mesg\_text); i++)

{

if (message.mesg\_text[i] != ' ')

{

tmp[cnt] = message.mesg\_text[i] - '0';

cnt++;

}

}

tmp[2] += '0';

switch (tmp[2])

{

case 43:

printf("%d + %d = %d\n", tmp[0], tmp[1], tmp[0] + tmp[1]);

break;

case 45:

printf("%d - %d = %d\n", tmp[0], tmp[1], tmp[0] - tmp[1]);

break;

case 120:

case 42:

printf("%d \* %d = %d\n", tmp[0], tmp[1], tmp[0] \* tmp[1]);

break;

case 47:

printf("%d / %d = %f\n", tmp[0], tmp[1], tmp[0] \* 1.0 / tmp[1]);

break;

}

// to destroy the message queue

msgctl(msgid, IPC\_RMID, NULL);

return 0;

}

Text

Description automatically generated

Bài 3:

**Message queue:**

**File: Writer.c**

// C Program for Message Queue (Writer Process)

#include <stdio.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <string.h>

// structure for message queue

struct mesg\_buffer

{

long mesg\_type;

char mesg\_text[100];

} message;

int main()

{

key\_t key;

int msgid;

// ftok to generate unique key

key = ftok("msg.txt", 1);

// msgget creates a message queue

// and returns identifier

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

message.mesg\_type = 1;

while(1){

printf("Write Data : ");

fgets(message.mesg\_text, sizeof(message.mesg\_text), stdin);

msgsnd(msgid, &message, sizeof(message), 0);

if(strcmp(message.mesg\_text, "exit\n") == 0)

break;

}

// msgsnd to send message

// display the message

return 0;

}

**File reader.c**

// C Program for Message Queue (Reader Process)

#include <stdio.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <string.h>

// structure for message queue

struct mesg\_buffer {

long mesg\_type;

char mesg\_text[100];

} message;

int main()

{

key\_t key;

int msgid;

// ftok to generate unique key

key = ftok("msg.txt",1);

// msgget creates a message queue

// and returns identifier

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

// msgrcv to receive message

// display the message

while(1){

msgrcv(msgid, &message, sizeof(message), 1, 0);

if(strcmp(message.mesg\_text, "exit\n") == 0)

break;

printf("Data received is : %s \n", message.mesg\_text);

}

// to destroy the message queue

msgctl(msgid, IPC\_RMID, NULL);

return 0;

}

Text

Description automatically generated

Bài thêm:

**Share memory:**

#include <stdio.h>

#include <unistd.h>

#include <limits.h>

#include <string.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include<time.h>

#define SIZE 256

int main(int argc, char\* argv[])

{

srand(time(NULL));

int i;

int \*shm, shmid, k,pid;

key\_t key;

if((key=ftok(".",65))==-1){

perror("Key created.\n");

return 1;

}

shmid = shmget(key, SIZE, IPC\_CREAT | 0666);

if (shmid == -1) {

perror("Shared memory created.\n");

return 2;

}

shm = (int\*) shmat(shmid, 0, 0);

pid = fork();

if(pid==0) { // child

FILE \*f = fopen("data", "w");

int n = atoi(argv[1]);

for (i = 0; i < n; ++i)

{

fprintf(f,"%d\n", rand() % 100);

}

fclose(f);

FILE \*f1 = fopen("data","r");

int k,x=1;

shm[0]=n;

shm[shm[0]+1]=-1;

while(fscanf(f1,"%d",&k) != EOF)

{

shm[x] = k;

x++;

}

fclose(f1);

//

sleep(3);

printf("Sum=%d\n",shm[shm[0]+1]);

printf("Mang sau khi sap xep:\n");

for (i = 1; i <= shm[0]; ++i)

{

printf("%d ",shm[i]);

}

shmdt((void\*) shm);

shmctl(shmid, IPC\_RMID, (struct shmid\_ds\*) 0);

return 0;

}

else if(pid >0) { // parent

sleep(1);

int sum=0;

for (i = 1; i <= shm[0]; ++i)

{

sum+=shm[i];

}

shm[shm[0]+1]=sum;

//sort

int j,k;

for (i = 1; i < shm[0]; ++i)

{

for (j = 1; j < shm[0]; ++j)

{

if (shm[i] < shm[j])

{

k = shm[i];

shm[i] = shm[j];

shm[j] = k;

}

}

}

//TODO

shmdt((void\*) shm);

sleep(5);

return 0;

}

else { perror("Fork failed."); return 4; }

return 0;

}

Text

Description automatically generated