**ЗАДАНИЕ №1** Подключить к очереди двух клиентов (один из них выполняется как поток).

**ЛИСТИНГ:**

**SERVER:**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

#include <sys/msg.h>

#define MAX\_TEXT 512

struct my\_msg\_st{

long int my\_msg\_type;

char some\_text[BUFSIZ];};

int main()

{

int running =1;

int msgid,msgid2;

struct my\_msg\_st some\_data;

char buffer[BUFSIZ];

msgid= msgget ((key\_t) 1234, 0666 | IPC\_CREAT);

msgid2= msgget ((key\_t) 1234, 0666 | IPC\_CREAT);

if (msgid==-1) { printf ("msget failed!");

getchar();

exit(EXIT\_FAILURE);}

while(running)

{printf ("enter some text:");

fgets(buffer, BUFSIZ, stdin);

some\_data.my\_msg\_type=1;

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

if (msgsnd(msgid,(void\*) & some\_data, MAX\_TEXT, 0)==-1)

{

printf("msgsnd failed!");

getchar();

exit(EXIT\_FAILURE);

}

if (msgsnd(msgid2,(void\*) & some\_data, MAX\_TEXT, 0)==-1)

{

printf("msgsnd failed!");

getchar();

exit(EXIT\_FAILURE);

}

printf("servermsg ended!");

getchar();

exit(EXIT\_SUCCESS);

}}

**CLIENT\_1:**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

#include <sys/msg.h>

struct my\_msg\_st {

long int my\_msg\_typr;

char some\_text[BUFSIZ];};

int main()

{

int running =1;

int msgid;

struct my\_msg\_st some\_data;

long int msg\_to\_receive=0;

msgid = msgget ((key\_t) 1234, 0666 | IPC\_CREAT);

if (msgid==-1)

{

printf ("msget failed!");

getchar();

exit(EXIT\_FAILURE);

}

while(running)

{if (msgrcv(msgid,(void\*) & some\_data, BUFSIZ, msg\_to\_receive,0)==-1){

printf("msgrcv failed!");

getchar();

exit(EXIT\_FAILURE);}

printf("Text captured:%s", some\_data.some\_text);

printf("client terminated");

getchar();

exit(EXIT\_SUCCESS);

}}

**CLIENT\_2**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

#include <sys/msg.h>

#include <pthread.h>

struct my\_msg\_st {

long int my\_msg\_typr;

char some\_text[BUFSIZ];};

void\* thread\_func(void\* arg);

char message[] ="date";

int main()

{

int res;

pthread\_t a\_thread;

void\* thread\_result;

res =pthread\_create(&a\_thread,NULL,thread\_func,(void\*)message);

res=pthread\_join(a\_thread, &thread\_result);

exit(0);

}

void \*thread\_func(void\* arg)

{

int running =1;

int msgid2;

struct my\_msg\_st some\_data;

long int msg\_to\_receive=0;

msgid2 = msgget ((key\_t) 1234, 0666 | IPC\_CREAT);

if (msgid2==-1)

{

printf ("msget failed!");

getchar();

exit(EXIT\_FAILURE);

}

while(running)

{if (msgrcv(msgid2,(void\*) & some\_data, BUFSIZ, msg\_to\_receive,0)==-1){

printf("msgrcv failed!");

getchar();

exit(EXIT\_FAILURE);}

printf("Text captured:%s", some\_data.some\_text);

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

{running=0;}

if (msgctl(msgid2, IPC\_RMID, 0)==-1){

printf("msgctl failed!");

getchar();

exit(EXIT\_FAILURE);}

printf("client terminated");

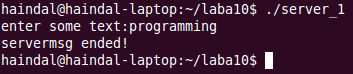
getchar();

exit(0);}

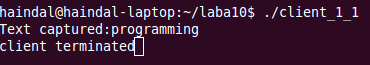
}

**ВЫПОЛНЕНИЕ:**

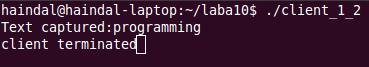
**SERVER:**

****

**CLIENT\_1**

****

**CLIENT\_2**

****

**ЗАДАНИЕ №2** Создать две очереди сообщений. Первый процесс пишет в первую очередь и читает из второй, второй процесс наоборот, читает из первой очереди и пишет во вторую.

**ЛИСТИНГ:**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

#include <sys/msg.h>

#define MAX\_TEXT 512

struct my\_msg\_st{

long int my\_msg\_type;

char some\_text[BUFSIZ];};

struct my\_msg\_st2{

long int my\_msg\_type2;

char some\_text2[BUFSIZ];};

int main()

{

pid\_t child\_pid;

child\_pid=fork();

int running =1;

int msgid;

int msgid2;

struct my\_msg\_st some\_data;

char buffer[BUFSIZ];

struct my\_msg\_st2 some\_data2;

char buffer2[BUFSIZ];

long int msg\_to\_receive=0;

msgid= msgget ((key\_t) 1234, 0666 | IPC\_CREAT);

msgid2= msgget ((key\_t) 1235, 0666 | IPC\_CREAT);

if (msgid==-1) { printf ("msget failed!");

getchar();

exit(EXIT\_FAILURE);}

if (msgid2==-1) { printf ("msget failed!");

getchar();

exit(EXIT\_FAILURE);}

if(child\_pid == 0)

{

printf ("enter some text1:");

fgets(buffer, BUFSIZ, stdin);

some\_data.my\_msg\_type=1;

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

if (msgsnd(msgid,(void\*) & some\_data, MAX\_TEXT, 0)==-1)

{

printf("msgsnd failed!");

exit(EXIT\_FAILURE);

}

if (msgrcv(msgid2,(void\*) & some\_data2, BUFSIZ, msg\_to\_receive,0)==-1)

{

printf("msgrcv failed!");

exit(EXIT\_FAILURE);

}

printf("Text captured2:%s", some\_data2.some\_text2);

if (msgctl(msgid2, IPC\_RMID, 0)==-1)

{

printf("msgctl failed!");

getchar();

exit(EXIT\_FAILURE);

}

}

else{

sleep(3);

printf ("enter some text2:");

fgets(buffer2, BUFSIZ, stdin);

some\_data2.my\_msg\_type2=1;

strcpy(some\_data2.some\_text2,buffer2);

if (msgsnd(msgid2,(void\*) & some\_data2, MAX\_TEXT, 0)==-1)

{

printf("msgsnd failed!");

exit(EXIT\_FAILURE);

}

if (msgrcv(msgid,(void\*) & some\_data, BUFSIZ, msg\_to\_receive,0)==-1)

{

printf("msgrcv failed!");

exit(EXIT\_FAILURE);

}

printf("Text captured1:%s", some\_data.some\_text);

if (msgctl(msgid, IPC\_RMID, 0)==-1){

printf("msgctl failed!");

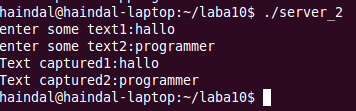
getchar();

exit(EXIT\_FAILURE);}

}

}

**ВЫПОЛНЕНИЕ:**

****

**ЗАДАНИЕ №3.** Сделать два сервера сообщений и одного клиента. Серверы используют одну и ту же очередь. Клиент читает и определяет, от какого сервера пришло сообщение.

**ЛИСТИНГ:**

**SERVER\_1:**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

#include <sys/msg.h>

#define MAX\_TEXT 512

struct my\_msg\_st{

long int my\_msg\_type;

char some\_text[BUFSIZ];};

int main()

{

int running =1;

int msgid;

struct my\_msg\_st some\_data;

char buffer[BUFSIZ];

msgid= msgget ((key\_t) 1234, 0666 | IPC\_CREAT);

if (msgid==-1) { printf ("msget failed!");

getchar();

exit(EXIT\_FAILURE);}

while(running)

{printf ("enter some text:");

fgets(buffer, BUFSIZ, stdin);

some\_data.my\_msg\_type=1;

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

if (msgsnd(msgid,(void\*) & some\_data, MAX\_TEXT, 0)==-1)

{

printf("msgsnd failed!");

getchar();

exit(EXIT\_FAILURE);

}

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

{

running =0;

}

printf("servermsg ended!");

getchar();

exit(EXIT\_SUCCESS);

}}

**SERVER\_2:**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

#include <sys/msg.h>

#define MAX\_TEXT 512

struct my\_msg\_st{

long int my\_msg\_type;

char some\_text[BUFSIZ];};

int main()

{

int running =1;

int msgid;

struct my\_msg\_st some\_data;

char buffer[BUFSIZ];

msgid= msgget ((key\_t) 1235, 0666 | IPC\_CREAT);

if (msgid==-1) { printf ("msget failed!");

getchar();

exit(EXIT\_FAILURE);}

while(running)

{printf ("enter some text:");

fgets(buffer, BUFSIZ, stdin);

some\_data.my\_msg\_type=1;

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

if (msgsnd(msgid,(void\*) & some\_data, MAX\_TEXT, 0)==-1)

{

printf("msgsnd failed!");

getchar();

exit(EXIT\_FAILURE);

}

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

{

running =0;

}

printf("servermsg ended!");

getchar();

exit(EXIT\_SUCCESS);

}}

**CLIENT:**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

#include <sys/msg.h>

struct my\_msg\_st {

long int my\_msg\_typr;

char some\_text[BUFSIZ];};

int main()

{

pid\_t child\_pid;

child\_pid=fork();

int running =1;

int msgid;

int msgid2;

struct my\_msg\_st some\_data;

long int msg\_to\_receive=0;

msgid = msgget ((key\_t) 1234, 0666 | IPC\_CREAT);

msgid2 = msgget ((key\_t) 1235, 0666 | IPC\_CREAT);

if (msgid==-1)

{

printf ("msget failed!");

getchar();

exit(EXIT\_FAILURE);

}

if(child\_pid == 0)

{

while(running)

{if (msgrcv(msgid,(void\*) & some\_data, BUFSIZ, msg\_to\_receive,0)==-1){

printf("msgrcv failed!");

getchar();

exit(EXIT\_FAILURE);}

printf("\nText captured server 1:%s", some\_data.some\_text);

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

{running=0;}

if (msgctl(msgid, IPC\_RMID, 0)==-1){

printf("msgctl failed!");

getchar();

exit(EXIT\_FAILURE);}

printf("client terminated");

getchar();

exit(EXIT\_SUCCESS);}

}

else {

while(running)

{

if (msgrcv(msgid2,(void\*) & some\_data, BUFSIZ, msg\_to\_receive,0)==-1)

{

printf("msgrcv failed!");

getchar();

exit(EXIT\_FAILURE);

}

printf("\nText captured server 2:%s", some\_data.some\_text);

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

{

running=0;

}

if (msgctl(msgid2, IPC\_RMID, 0)==-1)

{

printf("msgctl failed!");

getchar();

exit(EXIT\_FAILURE);

}

printf("client terminated");

getchar();

exit(EXIT\_SUCCESS);}

}

}