УО «Белорусский Государственный Технологический Университет»

**ФИТ**

Отчет по Лабораторной работе №10 на тему «Работа с очередью сообщений (Message Queue).»

Выполнили студенты

8-й группы:

Куницкий Н. Д.

Проверила:

Герман Ю. О

Минск 2016

**ЦЕЛЬ**. Изучить механизм очередей сообщений в LINUX.

**Краткие теоретические сведения**.

Очередь сообщений – это системный механизм, позволяющий выполнять обмен сообщениями между процессами. Условно, один процесс посылает в очередь сообщений текстовое сообщение (этот процесс играет роль сервера). Другие процессы могут считывать сообщения из очереди.

Для работы с очередью сообщений нужно ее создать (получить). Причем эту функцию должен выполнить как процесс-сервер, так и процесс-клиент. Данная функция имеет прототип

int msgget(key\_t key, int flags)

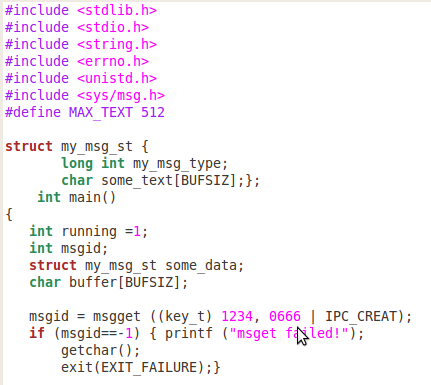
Очередь сообщений кодируется переменной key. Скажем так, это просто некоторое число, идентифицирующее очередь. Второй параметр flags задает атрибуты очереди. Например, можно объявить очередь закрытой для других процессов, кроме того, который ее создал. Хотя, разумеется, смысл в такой очереди не велик. Наконец, чтобы отправить сообщение в очередь используем функцию:

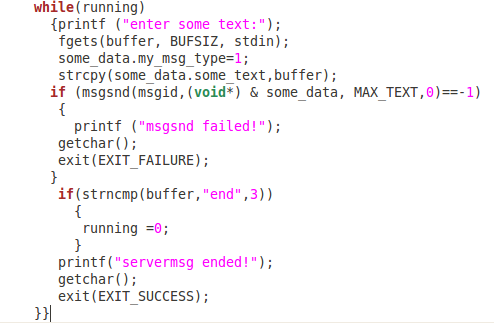
int msgsend(int id, const void\* msg\_ptr, size\_t msg\_size, int msg\_flags)

Здесь первый параметр задает номер очереди (возвращается предыдущей функцией). Второй параметр есть указатель на структуру данных с текстом сообщения. Третий параметр задает размеры сообщения. Четвертый – атрибуты сообщениия.

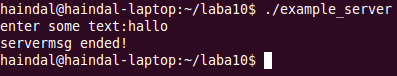
Итак, приведем приложение-сервер:

**//SERVER**



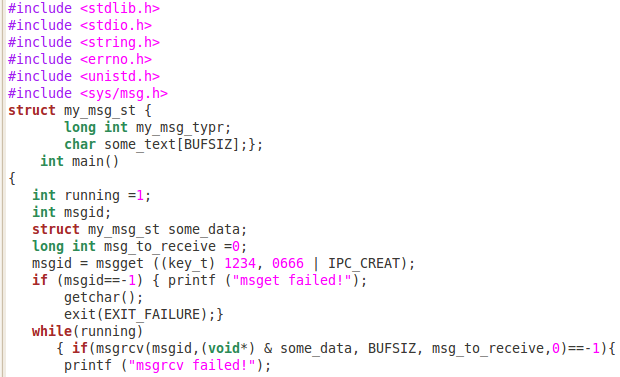


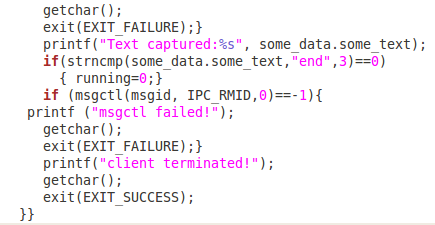
По команде msgget создаем очередь. Ее идентификатор присваивается переменной msgid. Атрибуты очереди: 0666 – доступная для чтения и записи, IPC\_CREAT – очередь создается «с нуля». Далее в цикле while просим ввести текст. Считываем его командой fgets. Вводимый текст попадает в буфер buffer. Командой strcpy переносим этот текст в поле some\_text структуры some\_data. Заносим сообщение в очередь командой msgsnd. Запустим это приложение:



Кстати, найдите то место в тексте кода, из-за которого программа досрочно выходит из цикла. Итак, сервер сработал. Очередь создана и находится в памяти. Теперь надо прочитать сообщение из очереди программой, которую условно назовем клиентом.

Клиенту также надо создать очередь, как это делал сервер. Потом ему надо прочитать сообщение. Для этого применяется команда msgrcv. Приведем текст:



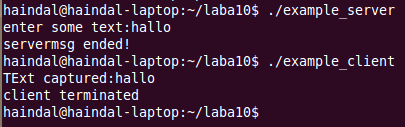


Сообщение считывается в поле some\_data.some\_text по команде

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

Переменная &some\_data представляет адрес структуры some\_data. Переменная msg\_to\_receive задает размер считанного из очереди сообщения в байтах.

Запустим клиента и получим:



**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");

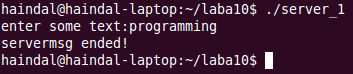
getchar();

exit(0);}

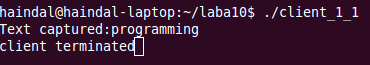
}

**ВЫПОЛНЕНИЕ:**

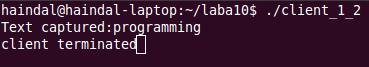
**SERVER:**

****

**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!");

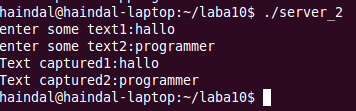
getchar();

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);}

}

}

**ВЫПОЛНЕНИЕ:**

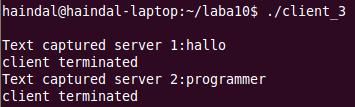
**SERVER\_1:**

****

**SERVER\_2:**

****

**CLIENT:**

****

**ВЫВОД:** Я изучил механизм очередей сообщений в LINUX.