**МИНОБРНАУКИ РОССИИ**

**САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра ВТ**

**ОТЧЕТ**

**по лабораторной работе №8**

**по дисциплине *«Организация процессов и программирование в среде Linux»***

**Тема: «ВЗАИМОДЕЙСТВИЕ ПРОЦЕССОВ**

**НА ОСНОВЕ СООБЩЕНИЙ»**

Cтудентка гр. 7306 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мищенко А.В.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Разумовский Г.В.

Санкт–Петербург

2020

**Цель работы:**

Знакомство с механизмом обмена сообщениями и системным вызовами приема и передачи сообщений.

**Задание:**

Написать две программы, обменивающиеся сообщениями. Первая

программа создает очередь и ожидает сообщение от второй программы определенное время, которое задается при запуске первой программы и выводится на экран. Если за это время сообщение от второй программы не поступило, то первая программа завершает свою работу и уничтожает очередь. Вторая программа может запускаться несколько раз и только при условии, что первая программа работает, в противном случае она заканчивает свою работу. При запуске второй программы указывается очередное время ожидания для первой программы.

**Ход работы:**

В ходе выполнения работы были написаны две программы: client.cpp и main.cpp. Исходные коды представлены в Приложении 1.

При запуске обоих приложений в качестве аргумента нужно передать время задержки: для приложения клиента - это значение будет передаваться серверу, если он на данный момент уже запущен, для приложения сервера - это значение ожидания, пока данное приложение не прекратит работу, если клиента не отправит ему новое значение. На рисунках 1 и 2 представлены запуски приложения клиента и сервера в разное время от друг друга.

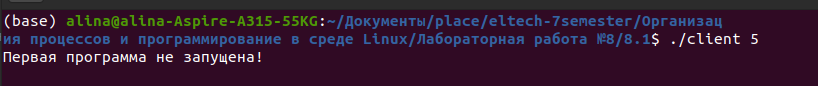


Рисунок 1. Попытка запустить приложение клиента без запущенного сервера



Рисунок 2. Запуск сервера на 5 секунд

На рисунках видно, что клиент не может быть запущен без работающего сервера, а сервер без клиента просто завершит свою работу через заданное время пользователем. На рисунках 3 и 4 приведен пример отправки сообщений от клиента к серверу.

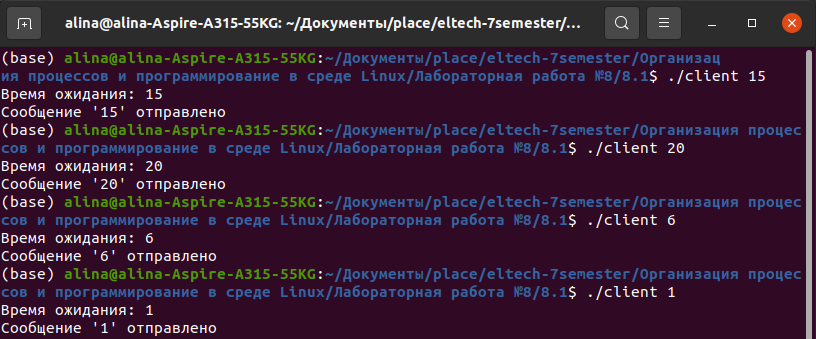


Рисунок 3. Запуск приложений клиента при открытом приложении сервера

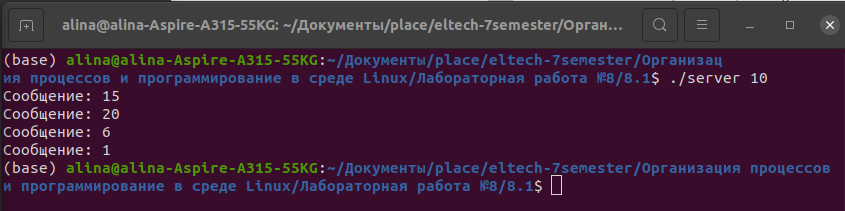


Рисунок 4. Работа сервера при запуске приложений клиента

**Вывод:**

В ходе лабораторной работы я научилась работать с механизмом обмена сообщениями между разными приложениями.

**Приложения**

Приложение 1.1. Файл main.cpp

#include <iostream>

#include <sys/msg.h>

#include <sys/ipc.h>

#include <sys/time.h>

#include <cstring>

#include <iomanip>

#include <sys/types.h>

#include <unistd.h>

#include <assert.h>

#include <signal.h>

using namespace std;

const int KEY = 646;

int msgId;

struct myMsgbuf {

long mtype; /\* тип сообщения, должен быть > 0 \*/

int mtext; /\* содержание сообщения \*/

};

void signal\_handler(int signum) {

if (signum == SIGALRM) {

msgctl(msgId, IPC\_RMID, 0);

exit(0);

}

}

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

assert(argc == 2);

signal(SIGALRM, signal\_handler);

int waitTime = atoi(argv[1]);

myMsgbuf message;

//разрешен прием сообщений пользователю, владеющему очередью и

//разрешена передача сообщений всем остальным пользователям

msgId = msgget(KEY, 402|IPC\_CREAT);

if(msgId == -1){

cout<<"Ошибка создания очереди!\n" << endl;

return 0;

}

struct itimerval value, ovalue;

// период перезапуска по умолчанию

value.it\_interval.tv\_sec = 0;

value.it\_interval.tv\_usec = 0;

// Первый запуск программы через waitTime сек

value.it\_value.tv\_sec = waitTime;

value.it\_value.tv\_usec = 0;

// Устанавливаем интервальный таймер

setitimer(ITIMER\_REAL, &value, &ovalue);

for(;;)

{

if( msgrcv(msgId, &message, sizeof(myMsgbuf), 0, 0) != -1){

cout<< "Сообщение: " << message.mtext << endl;

value.it\_value.tv\_sec = message.mtext;

setitimer(ITIMER\_REAL, &value, &ovalue);

}

}

}

Приложение 1.2. client.cpp

#include <iostream>

#include <sys/msg.h>

#include <sys/ipc.h>

#include <cstring>

#include <sys/types.h>

#include <unistd.h>

#include <assert.h>

using namespace std;

const int KEY = 646;

struct myMsgbuf {

long mtype; /\* тип сообщения, должен быть > 0 \*/

int mtext; /\* содержание сообщения \*/

};

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

assert(argc == 2);

int waitTime = atoi(argv[1]);

myMsgbuf message;

int msgId;

if((msgId = msgget(KEY, IPC\_CREAT|IPC\_EXCL))!= -1){

cout<< "Первая программа не запущена!\n"<<endl;

msgctl(msgId, IPC\_RMID, 0);

return 0;

}

msgId = msgget(KEY, IPC\_CREAT);

cout << "Время ожидания: " << waitTime << endl;

message.mtype = 1;

message.mtext = waitTime;

if( (msgsnd(msgId, &message, sizeof(myMsgbuf), 0)) != -1) {

cout<< "Сообщение '"<< message.mtext<< "' отправлено" << endl;

} else{

cout<< "Сообщение не отправлено" << endl;

perror("msgsnd");

}

return 0;

}