## МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

## «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

**Кафедра Вычислительной техники**

## ОТЧЕТ

**по лабораторной работе №8**

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

**Тема: Взаимодействие процессов на основе сообщений**

|  |  |
| --- | --- |
| Студент гр. 8308 | Петров Г.А. |
| Преподаватель | Разумовский Г.В. |

Санкт-Петербург 2021

# Цель работы

Целью лабораторной работы является знакомство с механизмом обмена сообщениями и системными вызовами приема и передачи сообщений.

# Задание

Написать три программы, выполняющиеся параллельно и читающие один и тот же файл. Программа, которая хочет прочитать файл, должна передать другим программам запрос на разрешение операции и ожидать их ответа. Эти запросы программы передают через одну очередь сообщений. Ответы каждая программа должна принимать в свою локальную очередь. В запросе указываются: номер программы, которой посылается запрос, идентификатор очереди, куда надо передать ответ, и время посылки запроса. Начать выполнять операцию чтения файла программе разрешается только при условии получения ответов от двух других программ. Каждая программа перед отображением файла на экране должна вывести следующую информацию: номер программы и времена ответов, полученных от других программ. Программа, которая получила запрос от другой программы, должна реагировать следующим образом:

* если программа прочитала файл, то сразу передается ответ, который должен содержать номер отвечающей программы и время ответа;
* если файл не читался, то ответ передается только при условии, что время посылки запроса в сообщении меньше, чем время запроса на чтение у данной программы.

Запросы, на которые ответы не были переданы, должны быть запомнены и после чтения файла обслужены.

# Примеры выполнения программ

Программы были разработаны и откомпилированы. После чего программы были запушена через три терминала. Результаты работы программ приведены на рисунках 1-4. Так же представлена распечатка файла TEXT.txt.

TEXT.txt

C:\Users\Admin\Desktop\Linuxshare\lab8\1.JPG

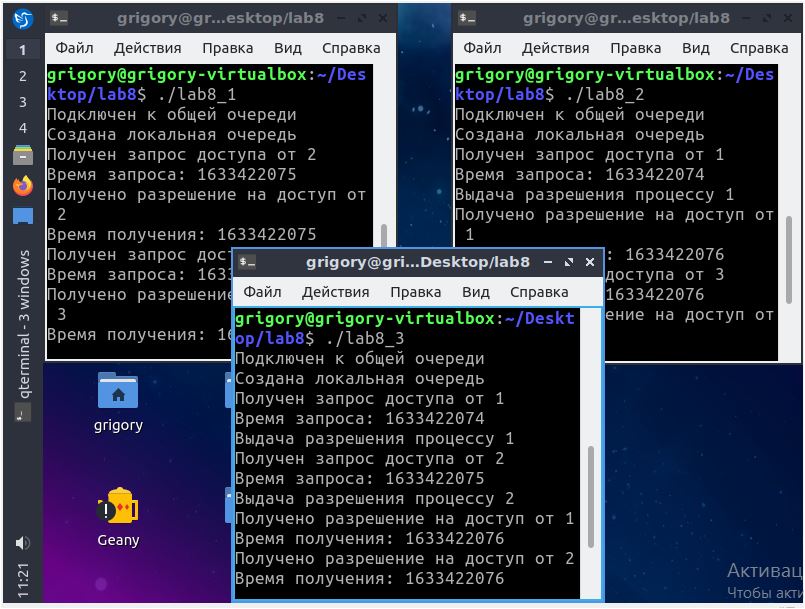


Рисунок 1

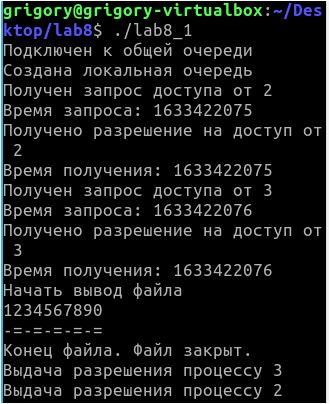


Рисунок 2

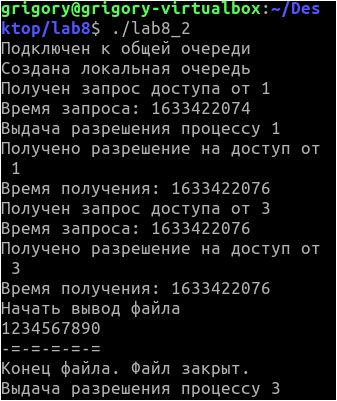


Рисунок 3

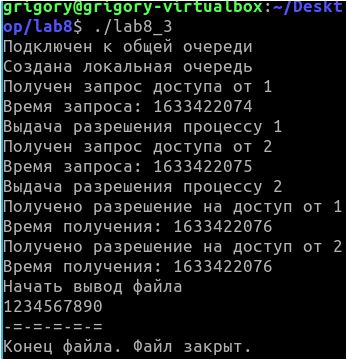


Рисунок 4

# Исходный код программ

## Lab8\_1.cpp

#include <iostream>

#include <fstream>

#include <sys/msg.h>

#define PID 1

#define FNAME "TEXT.txt"

//структура сообщений общей очереди

struct ask\_msg

{

long mtype;//тип сообщения

int sender;//отправитель

int answer\_queue;//локальная очередь заказчика

int ask\_time;//время запроса

};

//структура сообщений локальной очереди

struct answer\_msg

{

long mtype;//тип сообщения

int sender;//отправитель

};

int main()

{

bool is\_creator=false;

int proc\_end=0;

int local\_queue;

int joint\_queue;

ask\_msg ask;//отправка запроса

ask\_msg ask\_buf[2];//полученные запросы

int buf\_i=0;//количество запросов в ожидании

answer\_msg answer;//отправка ответов

answer\_msg access;//получение разрешений

int num\_get\_access=0;//количество полученных

int num\_send\_access=0;//количество отправленных

//создаем общую очередь

joint\_queue=msgget(777,0606|IPC\_CREAT|IPC\_EXCL);

if(joint\_queue!=-1)

{

is\_creator=true;

std::cout<<"Создана общая очередь"<<std::endl;

}

else

{//подключаемся если уже создана

joint\_queue=msgget(777,0606|IPC\_CREAT);

std::cout<<"Подключен к общей очереди"<<std::endl;

}

//создаем локальную очередь

local\_queue=msgget(IPC\_PRIVATE,0606|IPC\_CREAT);

std::cout<<"Создана локальная очередь"<<std::endl;

//инициализация запросов и занесение их в очередь

ask.ask\_time=time(NULL);

ask.mtype=(PID)%3+1;

ask.sender=PID;

ask.answer\_queue=local\_queue;

msgsnd(joint\_queue,&ask,sizeof(ask\_msg),0);

ask.mtype=(PID+1)%3+1;

msgsnd(joint\_queue,&ask,sizeof(ask\_msg),0);

//инициализация ответа

answer.mtype=1;

answer.sender=PID;

//ожидание двух разрешений на файл (проверка общей очереди)

while(num\_get\_access<2)

{

if(msgrcv(joint\_queue,&ask\_buf[buf\_i],sizeof(ask\_msg),PID,IPC\_NOWAIT)!=-1)

{//проверкa запросов в общей очереди для этой програмы

std::cout<<"Получен запрос доступа от "<<ask\_buf[buf\_i].sender<<std::endl;

std::cout<<"Время запроса: "<<ask\_buf[buf\_i].ask\_time<<std::endl;

if(ask\_buf[buf\_i].ask\_time<ask.ask\_time || (ask\_buf[buf\_i].ask\_time==ask.ask\_time && ask\_buf[buf\_i].sender<PID))

{//приоритет младшего и меньший индификатор

msgsnd(ask\_buf[buf\_i].answer\_queue,&answer,sizeof(answer\_msg),0);

++num\_send\_access;

std::cout<<"Выдача разрешения процессу "<<ask\_buf[buf\_i].sender<<std::endl;

}

else//запомнить в буфере если старше

++buf\_i;

}

if(msgrcv(local\_queue,&access,sizeof(answer\_msg),1,IPC\_NOWAIT)!=-1)

{//проверка разрешений в локальной очереди

++num\_get\_access;

std::cout<<"Получено разрешение на доступ от "<<access.sender<<std::endl;

std::cout<<"Время получения: "<<time(NULL)<<std::endl;

}

}

//вывод файла

std::cout<<"Начать вывод файла"<<std::endl;

std::ifstream fin(FNAME);

std::string str;

while(std::getline(fin,str))

std::cout<<str<<std::endl;

fin.close();

std::cout<<"Конец файла. Файл закрыт."<<std::endl;

//выдача разрешений всем ожидающим

while(buf\_i>0)

{

--buf\_i;

msgsnd(ask\_buf[buf\_i].answer\_queue,&answer,sizeof(answer\_msg),0);

++num\_send\_access;

std::cout<<"Выдача разрешения процессу "<<ask\_buf[buf\_i].sender<<std::endl;

}

//если еще не все запросили доступ

while(num\_send\_access<2)

{//проверка запросов из общей очереди для этой программы

if(msgrcv(joint\_queue,&ask\_buf[buf\_i],sizeof(ask\_msg),PID,IPC\_NOWAIT)!=-1)

{

msgsnd(ask\_buf[buf\_i].answer\_queue,&answer,sizeof(answer\_msg),0);

++num\_send\_access;

std::cout<<"Выдача разрешения процессу "<<ask\_buf[buf\_i].sender<<std::endl;

}

}

//отпрвака готовности завершения общей очереди

ask.mtype=0;

msgsnd(joint\_queue,&ask,sizeof(ask\_msg),0);

//ожиание готовности остальных процессов

if(is\_creator)

{

while(proc\_end<3)

{

if(msgrcv(joint\_queue,&ask,sizeof(ask\_msg),0,0)!=-1)

++proc\_end;

}

msgctl(joint\_queue,IPC\_RMID,0);

}

//удалене локальной очереди

msgctl(local\_queue,IPC\_RMID,0);

return 0;

}

## Lab8\_2.cpp

#include <iostream>

#include <fstream>

#include <sys/msg.h>

#define PID 2

#define FNAME "TEXT.txt"

struct ask\_msg

{

long mtype;

int sender;

int answer\_queue;

int ask\_time;

};

struct answer\_msg

{

long mtype;

int sender;

};

int main()

{

bool is\_creator=false;

int proc\_end=0;

int local\_queue;

int joint\_queue;

ask\_msg ask;//отправка запроса

ask\_msg ask\_buf[2];//полученные запросы

int buf\_i=0;//количество запросов в ожидании

answer\_msg answer;//отправка ответов

answer\_msg access;//получение разрешений

int num\_get\_access=0;//количество полученных

int num\_send\_access=0;//количество отправленных

//создаем общую очередь

joint\_queue=msgget(777,0606|IPC\_CREAT|IPC\_EXCL);

if(joint\_queue!=-1)

{

is\_creator=true;

std::cout<<"Создана общая очередь"<<std::endl;

}

else

{//подключаемся если уже создана

joint\_queue=msgget(777,0606|IPC\_CREAT);

std::cout<<"Подключен к общей очереди"<<std::endl;

}

//создаем локальную очередь

local\_queue=msgget(IPC\_PRIVATE,0606|IPC\_CREAT);

std::cout<<"Создана локальная очередь"<<std::endl;

//инициализация запросов и занесение их в очередь

ask.ask\_time=time(NULL);

ask.mtype=(PID)%3+1;

ask.sender=PID;

ask.answer\_queue=local\_queue;

msgsnd(joint\_queue,&ask,sizeof(ask\_msg),0);

ask.mtype=(PID+1)%3+1;

msgsnd(joint\_queue,&ask,sizeof(ask\_msg),0);

//инициализация ответа

answer.mtype=1;

answer.sender=PID;

//ожидание двух разрешений на файл (проверка общей очереди)

while(num\_get\_access<2)

{

if(msgrcv(joint\_queue,&ask\_buf[buf\_i],sizeof(ask\_msg),PID,IPC\_NOWAIT)!=-1)

{//проверкa запросов в общей очереди для этой програмы

std::cout<<"Получен запрос доступа от "<<ask\_buf[buf\_i].sender<<std::endl;

std::cout<<"Время запроса: "<<ask\_buf[buf\_i].ask\_time<<std::endl;

if(ask\_buf[buf\_i].ask\_time<ask.ask\_time || (ask\_buf[buf\_i].ask\_time==ask.ask\_time && ask\_buf[buf\_i].sender<PID))

{//приоритет младшего и меньший индификатор

msgsnd(ask\_buf[buf\_i].answer\_queue,&answer,sizeof(answer\_msg),0);

++num\_send\_access;

std::cout<<"Выдача разрешения процессу "<<ask\_buf[buf\_i].sender<<std::endl;

}

else//запомнить в буфере если старше

++buf\_i;

}

if(msgrcv(local\_queue,&access,sizeof(answer\_msg),1,IPC\_NOWAIT)!=-1)

{//проверка разрешений в локальной очереди

++num\_get\_access;

std::cout<<"Получено разрешение на доступ от "<<access.sender<<std::endl;

std::cout<<"Время получения: "<<time(NULL)<<std::endl;

}

}

//вывод файла

std::cout<<"Начать вывод файла"<<std::endl;

std::ifstream fin(FNAME);

std::string str;

while(std::getline(fin,str))

std::cout<<str<<std::endl;

fin.close();

std::cout<<"Конец файла. Файл закрыт."<<std::endl;

//выдача разрешений всем ожидающим

while(buf\_i>0)

{

--buf\_i;

msgsnd(ask\_buf[buf\_i].answer\_queue,&answer,sizeof(answer\_msg),0);

++num\_send\_access;

std::cout<<"Выдача разрешения процессу "<<ask\_buf[buf\_i].sender<<std::endl;

}

//если еще не все запросили доступ

while(num\_send\_access<2)

{//проверка запросов из общей очереди для этой программы

if(msgrcv(joint\_queue,&ask\_buf[buf\_i],sizeof(ask\_msg),PID,IPC\_NOWAIT)!=-1)

{

msgsnd(ask\_buf[buf\_i].answer\_queue,&answer,sizeof(answer\_msg),0);

++num\_send\_access;

std::cout<<"Выдача разрешения процессу "<<ask\_buf[buf\_i].sender<<std::endl;

}

}

//отпрвака готовности завершения общей очереди

ask.mtype=0;

msgsnd(joint\_queue,&ask,sizeof(ask\_msg),0);

//ожиание готовности остальных процессов

if(is\_creator)

{

while(proc\_end<3)

{

if(msgrcv(joint\_queue,&ask,sizeof(ask\_msg),0,0)!=-1)

++proc\_end;

}

msgctl(joint\_queue,IPC\_RMID,0);

}

//удалене локальной очереди

msgctl(local\_queue,IPC\_RMID,0);

return 0;

}

## Lab8\_3.cpp

#include <iostream>

#include <fstream>

#include <sys/msg.h>

#define PID 3

#define FNAME "TEXT.txt"

struct ask\_msg

{

long mtype;

int sender;

int answer\_queue;

int ask\_time;

};

struct answer\_msg

{

long mtype;

int sender;

};

int main()

{

bool is\_creator=false;

int proc\_end=0;

int local\_queue;

int joint\_queue;

ask\_msg ask;//отправка запроса

ask\_msg ask\_buf[2];//полученные запросы

int buf\_i=0;//количество запросов в ожидании

answer\_msg answer;//отправка ответов

answer\_msg access;//получение разрешений

int num\_get\_access=0;//количество полученных

int num\_send\_access=0;//количество отправленных

//создаем общую очередь

joint\_queue=msgget(777,0606|IPC\_CREAT|IPC\_EXCL);

if(joint\_queue!=-1)

{

is\_creator=true;

std::cout<<"Создана общая очередь"<<std::endl;

}

else

{//подключаемся если уже создана

joint\_queue=msgget(777,0606|IPC\_CREAT);

std::cout<<"Подключен к общей очереди"<<std::endl;

}

//создаем локальную очередь

local\_queue=msgget(IPC\_PRIVATE,0606|IPC\_CREAT);

std::cout<<"Создана локальная очередь"<<std::endl;

//инициализация запросов и занесение их в очередь

ask.ask\_time=time(NULL);

ask.mtype=(PID)%3+1;

ask.sender=PID;

ask.answer\_queue=local\_queue;

msgsnd(joint\_queue,&ask,sizeof(ask\_msg),0);

ask.mtype=(PID+1)%3+1;

msgsnd(joint\_queue,&ask,sizeof(ask\_msg),0);

//инициализация ответа

answer.mtype=1;

answer.sender=PID;

//ожидание двух разрешений на файл (проверка общей очереди)

while(num\_get\_access<2)

{

if(msgrcv(joint\_queue,&ask\_buf[buf\_i],sizeof(ask\_msg),PID,IPC\_NOWAIT)!=-1)

{//проверкa запросов в общей очереди для этой програмы

std::cout<<"Получен запрос доступа от "<<ask\_buf[buf\_i].sender<<std::endl;

std::cout<<"Время запроса: "<<ask\_buf[buf\_i].ask\_time<<std::endl;

if(ask\_buf[buf\_i].ask\_time<ask.ask\_time || (ask\_buf[buf\_i].ask\_time==ask.ask\_time && ask\_buf[buf\_i].sender<PID))

{//приоритет младшего и меньший индификатор

msgsnd(ask\_buf[buf\_i].answer\_queue,&answer,sizeof(answer\_msg),0);

++num\_send\_access;

std::cout<<"Выдача разрешения процессу "<<ask\_buf[buf\_i].sender<<std::endl;

}

else//запомнить в буфере если старше

++buf\_i;

}

if(msgrcv(local\_queue,&access,sizeof(answer\_msg),1,IPC\_NOWAIT)!=-1)

{//проверка разрешений в локальной очереди

++num\_get\_access;

std::cout<<"Получено разрешение на доступ от "<<access.sender<<std::endl;

std::cout<<"Время получения: "<<time(NULL)<<std::endl;

}

}

//вывод файла

std::cout<<"Начать вывод файла"<<std::endl;

std::ifstream fin(FNAME);

std::string str;

while(std::getline(fin,str))

std::cout<<str<<std::endl;

fin.close();

std::cout<<"Конец файла. Файл закрыт."<<std::endl;

//выдача разрешений всем ожидающим

while(buf\_i>0)

{

--buf\_i;

msgsnd(ask\_buf[buf\_i].answer\_queue,&answer,sizeof(answer\_msg),0);

++num\_send\_access;

std::cout<<"Выдача разрешения процессу "<<ask\_buf[buf\_i].sender<<std::endl;

}

//если еще не все запросили доступ

while(num\_send\_access<2)

{//проверка запросов из общей очереди для этой программы

if(msgrcv(joint\_queue,&ask\_buf[buf\_i],sizeof(ask\_msg),PID,IPC\_NOWAIT)!=-1)

{

msgsnd(ask\_buf[buf\_i].answer\_queue,&answer,sizeof(answer\_msg),0);

++num\_send\_access;

std::cout<<"Выдача разрешения процессу "<<ask\_buf[buf\_i].sender<<std::endl;

}

}

//отпрвака готовности завершения общей очереди

ask.mtype=0;

msgsnd(joint\_queue,&ask,sizeof(ask\_msg),0);

//ожиание готовности остальных процессов

if(is\_creator)

{

while(proc\_end<3)

{

if(msgrcv(joint\_queue,&ask,sizeof(ask\_msg),0,0)!=-1)

++proc\_end;

}

msgctl(joint\_queue,IPC\_RMID,0);

}

//удалене локальной очереди

msgctl(local\_queue,IPC\_RMID,0);

return 0;

}

# Вывод

При выполнении лабораторной работы изучены и использованы механизмы обмена сообщениями и системные вызовы приема и передачи сообщений. Программа разработанная в соответствии с заданием, работает корректно.