Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ОТЧЕТ ПО ДИСЦИПЛИНЕ

«Операционная система Linux»

ЛАБОРАТОРНАЯ РАБОТА №9

«Очереди сообщений»

Выполнили студенты группы 19ВВ2:

Гусев Д.О.

Кубасов И.М.

Приняли:

Федюнин Р.Н.

Семенов А.О.

Пенза 2022

**Цель работы :** Изучение механизмов взаимодействия процессов с использованием очередей сообщений.

Вариант 2.

**Задания:** Реализовать цепочку процессов, в которой каждый принимает через очередь сообщений данные только от процесса с номером на 1 больше своего. При подключении каждый новый процесс должен поместить в очередь сообщений информацию о себе (порядковый номер и свой PID). Номер процесса передавать в качестве аргумента при запуске приложения. Первый запущенный процесс должен выводить на терминал информацию о подключившихся процессах, поступающую из очереди сообщений. При завершении основного процесса удалить очередь сообщений и все остальные процессы должны завершиться.

**Листинг программы:**

#include <iostream>

#include <cstdlib>

#include <sys/stat.h>

#include <fcntl.h>

#include <signal.h>

# include <sys/types.h>

# include <sys/ipc.h>

#include <sys/msg.h>

#include <unistd.h>

#include <string.h>

#include <pthread.h>

using namespace std;

#define Proj\_Id 13

int IDmsq = 0; //идентификатор очереди сообщений

const char\* MSG\_QUEUE = "/tmp/msg\_queue1";

struct mymsgbuf

{

long mtype;

char mtext[256];

} ;

void sigSIGINT(int sig)

{

msqid\_ds \*buf;

char command[]="IPC\_RMID";

int a=msgctl(IDmsq, IPC\_RMID,buf);

cout<<"Delete"<<endl;

raise(4); //SIGILL

}

void\* threadFuncSend(void\* thread\_data)

{

for (;;)

{

int a=msgsnd(IDmsq,thread\_data,256,IPC\_NOWAIT);

sleep(1);

}

}

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

{

remove(MSG\_QUEUE);

int file = open(MSG\_QUEUE, O\_CREAT | O\_RDWR | O\_APPEND, 0755);

close(file);

int Number=atoi(argv[1]);

key\_t key;

if (Number==1)

{

signal(SIGINT, sigSIGINT);

key=ftok(MSG\_QUEUE, Proj\_Id);

if(key==-1)

cout<<"Error Ftok"<<endl;

IDmsq=msgget(key,IPC\_CREAT|0660); //0660 необходим для разрешения записи и чтения

if(IDmsq==-1)

cout<<"Error Msgget"<<endl;

mymsgbuf \*myBuf=new mymsgbuf;

for(;;)

{

msgrcv(IDmsq,myBuf,256,1,0);

cout<<"jjj"<<myBuf->mtext<<endl;

}

}

else

{

key=ftok(MSG\_QUEUE, Proj\_Id);

if(key==-1)

cout<<"Error Ftok"<<endl;

IDmsq=msgget(key,IPC\_CREAT);

if(IDmsq==-1)

cout<<"Error Msgget"<<endl;

pid\_t tid;

tid = gettid();

char Help[64];

sprintf (Help, "%d", tid);

char Pid[32]=" Pid Procces: ";

strcat(Pid,Help);

char NumberProcces[32]=" Number Procces: ";

strcat(NumberProcces,argv[1]);

int errno;

mymsgbuf \*SendBuf=new mymsgbuf;

SendBuf->mtype=1;

strcat(SendBuf->mtext,NumberProcces);

strcat(SendBuf->mtext,Pid);

mymsgbuf \*ReceiveBuf=new mymsgbuf;

ReceiveBuf->mtype=Number+1;

msgsnd(IDmsq,SendBuf,256,IPC\_NOWAIT);

SendBuf->mtype=Number;

pthread\_t thread;

pthread\_create(&thread, NULL, threadFuncSend, (void\*)SendBuf);

for(;;)

{

msgrcv(IDmsq,ReceiveBuf,256,Number+1,0);

if (errno==EIDRM)

{

pthread\_cancel(thread);

return 0;

}

cout<<"Message:"<<ReceiveBuf->mtext<<endl;

}

}

return 0;

}

**Результат работы программы:**

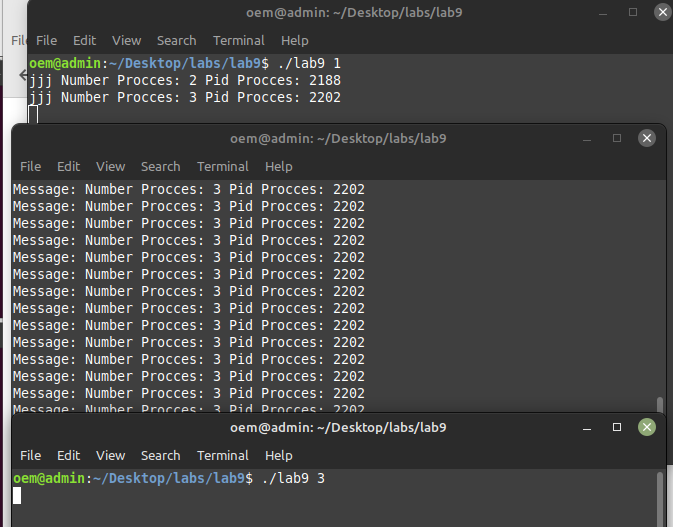


Рисунок 1— Взаимодействие процессов

**Вывод:** В ходе выполнения лабораторной работы были изучены способы передачи данных между процессами с использованием очереди сообщений. Была написана программа которая взаимодействует с очередью сообщений.