Лабораторная работа №8. Очереди сообщений UNIX

Цели работы

1. Получение навыков при работе с очередями.

Используемое программное обеспечение

При выполнении лабораторной работы будет использовано следующие про-граммное обеспечение: gcc – компилятор C/C++, vi – текстовый редактор.

Порядок выполнения лабораторной работы

1. Войти в систему Linux, указав имя и пароль, предварительно получив их у администратора.
2. В каталоге вашей группы создать файл программу с расширением *.cpp по варианту.
3. Написать программу на тему «Работа с очередями сообщений в Unix».
4. Собрать программу.
5. Исполнить программу

Пример работы с сообщениями Unix

Пересылка сообщения между двумя процессами.

Создание очереди сообщений и посылка простого сообщения

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <iostream>
#include <string.h>

#define MSGSZ     128

/*
 * Определение структуры сообщения.
 */
using namespace std;
typedef struct msgbuf
{
         long    mtype;
         char    mtext[MSGSZ];
 } message_buf;

void main()
{
    int msqid;
    int msgflg = IPC_CREAT | 0666;
    key_t key;
    message_buf sbuf;
    size_t buf_length;

    /*
     * Получаем id очереди сообщений 1234
     */
    key = 1234;

    cerr<<"\nmsgget: Calling msgget("<<key<<","<<msgflg<<")\n";

    if((msqid = msgget(key, msgflg )) < 0)
    {
        perror("msgget");
        exit(1);
    }
    else
    cerr<<"msgget: msgget succeeded: msqid = "<<msqid<<"\n";
    /*
```

```
     * Будем посылать сообщение с типом 1
     */

    sbuf.mtype = 1;

    cerr<<"msgget: msgget succeeded: msqid = "<<msqid<<"\n";

    strcpy(sbuf.mtext, "Did you get this?");

    cerr<<"msgget: msgget succeeded: msqid = "<<msqid<<"\n";

    buf_length = strlen(sbuf.mtext) + 1 ;

     /*
     * Посылаем сообщение.
     */
    if(msgsnd(msqid, &sbuf, buf_length, IPC_NOWAIT) < 0)
    {
    printf("%d, %d, %s, %d\n", msqid, sbuf.mtype, sbuf.mtext, buf_length);
        perror("msgsnd");
        exit(1);
    }
    else
    cout<<"Message: \""<<sbuf.mtext<<"\" Sent\n";

    exit(0);
}
```

Прием простого сообщения переданного сервером

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <iostream>

#define MSGSZ     128

using namespace std;
/*
 * Определение структуры сообщения.
 */

typedef struct msgbuf {
    long    mtype;
    char    mtext[MSGSZ];
} message_buf;


void main()
{
    int msqid;
    key_t key;
    message_buf  rbuf;

    /*
     * Получаем id очереди сообщений 1234
     */
    key = 1234;

    if((msqid = msgget(key, 0666)) < 0)
    {
        perror("msgget");
        exit(1);
    }


    /*
     * Получаем сообщение с типом 1.
     */
```

```
    if(msgrcv(msqid, &rbuf, MSGSZ, 1, 0) < 0)
    {
        perror("msgrcv");
        exit(1);
    }
    /*
     * Печатаем ответ.
     */
    cout<<rbuf.mtext<<"\n";
    exit(0);
}
```

Варианты задания на лабораторную работу

1. Написать 2 программы, которые обе одновременно будут принимать, и передавать сообщения и осуществить между ними следующий диалог:
   • (Процесс 1) Передает сообщение “Are you hearing me?”
   • (Процесс 2) Принимает сообщение и отвечает “Loud and Clear”.
   • (Процесс 1) Принимает ответ и передает “I can hear you too”.
2. Написать серверную и клиентскую программы, такие, что сервер может общаться с каждым клиентом индивидуально, используя при этом одну очередь сообщений.
3. Реализовать блокирующий или синхронный метод передачи сообщений между процессами, используя сигналы.
4. Написать локальный чат, состоящий из двух программ: сервера работающего на одной консоли и клиента, работающего на другой консоли одного компьютера.

Требования к отчёту по лабораторной работе

Отчёт должен содержать:

1. Титульный лист
2. Вариант задания
3. Блок-схему и листинг программы.
4. Вывод