**Постановка задачи**

Организовать взаимодействие типа клиент-сервер. Клиент делает запрос серверу о передаче файлов с определенным расширением из указанной директории. Сервер сканирует указанную директорию и отправляет клиенту список файлов, удовлетворяющих запросу.

**Описание метода решения задачи**

Сервер:

1. Создать сокет;
2. Связать сетевой адрес компьютера с идентификатором сокета;
3. Включить прием соединений;
4. Создать дополнительный сокет для работы с соединением;
5. Ожидать запрос на соединение от клиента;
6. После получения запроса, чтение данных из сокета;
7. Поиск в директории файлов с данной маской;
8. Послать клиенту имена найденных файлов;
9. Закрыть сокеты.

Клиент:

1. Создать сокет;
2. Выполнения запроса на подключение к серверному процессу;
3. Ввод директории;
4. Если директория не найдена – возврат на п. 3;
5. Посылка данных серверу;
6. Чтение данных от сервера;
7. Вывод данных в файл;
8. Закрыть сокет.

**Описание программного средства**

Программа написана на языке C++. Для получения исполняемого файла исходный текст программы следует скомпилировать каким-либо компилятором языка C++. Исполняемый файл может работать под любой UNIX-совместимой ОС, поддерживающей соответствующий формат исполняемого файла.

Запуск программы-сервера осуществляется командой:

./serv ,

2 части

./clnt

Запускаются без параметров.

**Исходный текст**

Файл lab8\_server.cpp

#include <iostream>

#include <sys/types.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <cstdlib>

#include <netinet/in.h>

#include <unistd.h>

#include <stdio.h>

using namespace std;

int main()

{

int sockid, sockid1; // socked identificator

struct sockaddr st\_addr;

char path[30], mask[30];

char data[1000];

int pipeFd[2];

remove("INADDR\_ANY");

if ((sockid = socket(AF\_UNIX,SOCK\_STREAM,0)) == -1) // creating socket

cout << "Server: Error in creating socket." << endl, exit(1);

//initialization of struct sockaddr

st\_addr.sa\_family = AF\_UNIX;

sprintf(st\_addr.sa\_data,"%s", "INADDR\_ANY");

// binding socket

if(bind(sockid, &st\_addr, sizeof(struct sockaddr))==-1)

{

cout << "Server: Error binding." << endl;

close(sockid);

exit(1);

}

// turn on TCP-connections

cout << "Server: is trying to listen socket with id = " << sockid << endl;

if(listen(sockid, 10)==-1)

{

cout<<"Error listening for server1!"<<endl;

close(sockid);

exit(1);

}

cout << "Server: Listening." << endl;

cout << "Server: Creating another socket for receiving messages." << endl;

sockid1 = accept(sockid, 0, 0);

cout << "Server: Created another socket with id = " << sockid1 << endl;

cout << "Server: Waiting for a message from client." << endl;

if(recv(sockid1, path, 30\*sizeof(char), MSG\_DONTROUTE))

cout << "Server: Received from client info: " << path << endl;

if(recv(sockid1, mask, 30\*sizeof(char), MSG\_DONTROUTE))

cout<<"Server: Received from client info: " << mask << endl;

pipe(pipeFd);

if (!fork())

{

close(pipeFd[0]);

close(1);

dup(pipeFd[1]);

close(pipeFd[1]);

execlp("find", "find",path,"-name", mask, "-print",NULL);

exit(0);

}

close(pipeFd[1]);

for (int i = 0; ; i++)

{

if (read(pipeFd[0], &data[i], 1) == 0)

{

data[i] = NULL;

break;

}

}

if(send(sockid1,data,sizeof(data)-10,MSG\_WAITALL)!=-1)

cout << "Server: Send data." << endl;

cout << "Server: Closing socket." << endl;

close(sockid1);

close(sockid);

cout << "Server: Closed socket." << endl;

return 0;

}

Файл lab8\_client.cpp

#include <iostream>

#include <fstream>

#include <sys/types.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <cstdlib>

#include <netinet/in.h>

#include <unistd.h>

#include <dirent.h>

#include <stdio.h>

using namespace std;

ofstream out;

int main()

{ int sockid;

sockaddr st\_addr;

in\_addr addr;

char mask[50], path[30], buf[1000];

//create socket

cout << "Client: creating socket." << endl;

if((sockid = socket(AF\_UNIX, SOCK\_STREAM, 0))==-1)

cout << "Client: Can't create socket." << endl, exit(1);

cout<<"Client: succsessfully created socket."<<endl;

//initialization of struct sockaddr & in\_addr

st\_addr.sa\_family = AF\_UNIX;

addr.s\_addr = INADDR\_ANY;

sprintf(st\_addr.sa\_data,"%s", "INADDR\_ANY");

//connection to server

cout << "Client: Connecting to server." << endl;

if (connect(sockid, &st\_addr, sizeof(struct sockaddr))==-1)

{

cout << "Client: Error conecting server." << endl;

close(sockid);

exit(1);

}

cout << "Client: Sucsessfully connected to server." << endl;

//ask for file path & mask

in\_path: cout << "Input file path: " << endl;

cin >> path;

if(!(opendir(path))) {cout << "It's not directory" << endl; goto in\_path;}

cout << "Input file mask: " << endl;

cin >> mask;

//sending to server information

cout << "Client: Sending to server search file info." << endl;

if (send(sockid, path, 30\*sizeof(char),MSG\_WAITALL)==-1) // send path

{

cout << "Client: Error sending info." << endl;

close(sockid);

exit(1);

}

if (send(sockid, mask, 30\*sizeof(char),MSG\_WAITALL)==-1) // send mask

{

cout << "Client: Error sending info." << endl;

close(sockid);

exit(1);

}

cout << "Client: Send file info sucsessfully." << endl;

if ((recv(sockid, buf, 1000\*sizeof(char), 0))!=-1) // receive file names

cout << "Client: Received from server info. " << endl;

out.open("output.txt");

if(!out) cout << "Error in open file 'output.txt'" << endl;

out << "Path is " << path << endl;

out << "Mask is " << mask << endl;

out << buf;

close(sockid);

cout << "Client: Closed socket." << endl;

}

**Пример выполнения**

**Тест 1.**

Запуск сервера.

Запуск клиента, ввод данных:

директория: .

Маска: \*.cpp

Выходной файл:

Path is .

Mask is \*.cpp

./lab8\_client.cpp

./lab8\_server.cpp

Сообщения от сервера:

Server: is trying to listen socket with id = 3

Server: Listening.

Server: Creating another socket for receiving messages.

Server: Created another socket with id = 4

Server: Waiting for a message from client.

Server: Received from client info: .

Server: Received from client info: \*.cpp

Server: Send data.

Server: Closing socket.

Server: Closed socket.

Сообщения от клиента:

Client: creating socket.

Client: succsessfully created socket.

Client: Connecting to server.

Client: Sucsessfully connected to server.

Input file path:

.

Input file mask:

\*.cpp

Client: Sending to server search file info.

Client: Send file info sucsessfully.

Client: Received from server info.

Client: Closed socket.

**Тест 2.**

Запуск клиента без запуска сервера.

Вывод на экран:

Client: creating socket.

Client: succsessfully created socket.

Client: Connecting to server.

Client: Error conecting server.