Постановка задачи:

Организовать взаимодействие типа клиент-сервер. Клиент подключается к первому серверу и передает запрос на получение определенного файла. Если этого файла нет, сервер подключается ко второму серверу и ищет файл там. Затем либо найденный файл пересылается клиенту, либо высылается сообщение, что такого файла нет.

Описание метода решения задачи

Данная задача решена посредством организации клиент-серверного взаимодействия. Клиент посылает серверу сообщение, содержащее имя файла. Сервер реализует поиск данного файла. При его отсутствии первый сервер посылает запрос второму. Таким образом между ними организуется клиент-серверная связь, то есть первый сервер становится клиентом по отношению ко второму. В свою очередь второй сервер ищет нужный файл, затем сообщает о результатах поиска первому серверу – своему клиенту, а тот извещает своего клиента о наличии или отсутствии файла. Также пользователю выдается сообщение о том, на каком именно сервере найден искомый файл. Серверы эмулируются двумя директориями, в которых осуществляется поиск заданного файла.

Организация клиент-серверного взаимодействия включает в себя следующие обязательные этапы:

**алгоритм сервера:**

* Создание сокета - socket();
* Связь адреса сервера с сокетом – bind();
* Включения приёма TCP-соединения – listen();
* Установка соединения с клиентом; - accept()
* Создание дочернего процесса – fork();
* Приём и посылка данных клиенту - recv(), send();
* Закрытие TCP-соединения – close();

**алгоритм клиента:**

* Создание сокета - socket();
* Подключение к серверу, связав сокет с адресом сервера – connect();
* Посылка и приём данных от сервера – send(), recv();
* Закрытие TCP-соединения – close();

Описание программного средства

Для реализации поставленной задачи использованы следующие программные средства:

* int bind (int sockfd, const struct sockaddr \*address, size\_t add\_len) - связывает сетевой

адрес компьютера с идентификатором сокета.

* int listen (int sockfd, int queue\_size) - включение приема TCP-соединений.
* int accept (int sockfd, struct sockaddr \*address, size\_t \*add\_len) - прием запроса на установку TCP-соединения, дополнительный сокет для работы с соединением.
* int connect (int csockfd, const struct sockaddr \*address, size\_t add\_len) - подключение клиента, выполняет запрос на подключение к серверному процессу.
* ssize\_t recv (int sockfd, void \*buffer, size\_t length, int flags) – используется для пересылки данных между клиентом и сервером (чтение данных).
* ssize\_t send (int sockfd, const void \*buffer, size\_t length, int flags) - используется для пересылки данных между клиентом и сервером (посылает данные).
* close() - закрытие TCP-соединения.

Исходный текст

/\*\*

\* Программа клиент-серверного взаимодействия

\*

\*@author Buzmakova

\* @author Zubakova

\*@version 1.0

\*/

#include <iostream> /\*подключение стандартных библиотек для работы\*/

#include <sys/types.h>

#include <sys/socket.h>

#include <fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <wait.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#include <dirent.h>

using namespace std;

/\*\*

\* проверка корректности соответствующего системного вызова

\*

\*@ param p результат системного вызова

\*@ param key номер, соответствующий системному вызову, необходимый для

\*распознавания ошибки вызова

\*@ param serv\_1 сокет для первого сервера

\*@ param serv\_2 сокет для второго сервера

\*/

void proverka(int p, int key, struct sockaddr serv\_1, struct sockaddr serv\_2)

{

if(p < 0)

switch(key)

{ /\* в случае ошибки удаляем из каталога точку входа для файла (unlink), чтобы это имя было доступно для работы в дальнейшем, выходим в случае ошибки\*/

case 1:{ perror("ошибка SOKET\n"); unlink (serv\_1.sa\_data);

unlink (serv\_2.sa\_data); exit(1);}

case 2:{ perror("ошибка FORK\n"); unlink (serv\_1.sa\_data);

unlink (serv\_2.sa\_data); exit(1);}

case 3:{ perror("ошибка CONNECT\n"); unlink (serv\_1.sa\_data);

unlink (serv\_2.sa\_data); exit(1);}

case 4:{ perror("ошибка SEND\n"); unlink (serv\_1.sa\_data);

unlink (serv\_2.sa\_data); exit(1);}

case 5:{ perror("ошибка RECV\n"); unlink (serv\_1.sa\_data);

unlink (serv\_2.sa\_data); exit(1);}

case 6:{ perror("ошибка BIND\n"); unlink (serv\_1.sa\_data);

unlink (serv\_2.sa\_data); exit(1);}

case 7:{ perror("ошибка LISTEN\n"); unlink (serv\_1.sa\_data);

unlink (serv\_2.sa\_data); exit(1);}

case 8:{ perror("ошибка READ\n"); unlink (serv\_1.sa\_data);

unlink (serv\_2.sa\_data); exit(1);}

case 9:{ perror("ошибка OPEN\n"); unlink (serv\_1.sa\_data);

unlink (serv\_2.sa\_data); exit(1);}

}

}

/\*\*

\* поиск файла на сервере

\*

\*@ param serv номер сервера

\*@ param sock\_work идентификатор сокета для связи

\*@ param \*file имя искомого файла

\*@ param serv\_1 сокет для первого сервера

\*@ param serv\_2 сокет для второго сервера

\*@ return если искомый файл найден, то возвращает 1

\*/

int poisk(int serv, int sock\_work, char \*file, struct sockaddr &serv\_1, struct sockaddr &serv\_2)

{

DIR \*d; /\* дескроиптор каталога \*/

char str[10]=" "; /\* для хранения имени директории \*/

char str1[10]=" "; /\* для хранения данных, прочитанных из файла \*/

int fd, flag = 0;

struct stat ss; /\* для хранения информации о файле \*/

struct dirent \*dir; /\* для хранения записей каталога \*/

if (serv == 1) strcpy(str, "server\_11"); /\*по номеру сервера определяем, где искать

if (serv == 2) strcpy(str, "server\_22"); файл \*/

d = opendir(str); /\* открытие каталога \*/

dir = readdir(d); /\* считываем первую запись каталога \*/

for(; dir!=NULL; dir=readdir(d))

{ /\* считываем очередную запись каталога в поисках заданного файла \*/

stat(dir->d\_name, &ss); /\* помещаем информацию обочередном файле в ss \*/

if ((strcmp(dir -> d\_name, file)) == 0) /\* сравниваем имя искомого файла и очередного файла из каталога \*/

{ /\* если совпали имена, то считываем данные из файла и передаем клиенту \*/

flag = 1;

chdir(str); /\* заходим в дирректорию \*/

/\* открытие файла, проверка на корректность вызова open \*/

proverka((fd = open(file, O\_RDONLY)), 9, serv\_1, serv\_2);

/\* считываем данные из фаела, проверка на корректность read \*/

proverka(read(fd, &str1, 20\*sizeof(char)), 8, serv\_1, serv\_2);

/\* отсылаем данные клиенту

proverka(send(sock\_work, str1 , 20\*sizeof(str), 0), 4, serv\_1, serv\_2);

}

}

strcpy(str, "~/xxx");

chdir(str); /\* возвращаемся в рабочую директорию \*/

return flag; /\* возвращаем результат поиска \*/

}

/\*\*

\* функция клиента

\*

\*@ param number номер сервера

\*@ param \*file имя искомого файла

\*@ param server сокет для первого сервера

\*@ param server \_ сокет для второго сервера

\*/

void client\_(int number, char \*file, struct sockaddr &server, struct sockaddr &server\_)

{

int sock,i,fd;

char str[40] = " "; /\* буфер для хранения полученных данных \*/

/\* создание сокета для связи, проверка корретности вызова socket \*/

proverka((sock = socket (AF\_UNIX, SOCK\_STREAM, 0)), 1, server, server\_);

/\* отправляем запрос серверу на подключение, ждем ответа \*/

while (connect (sock, &server, sizeof(struct sockaddr)) == -1)

cout <<"Ошибка подключения к серверу\n";

/\* после установки связи отсылаем серверу запрос с именем искомого файла \*/

proverka(send (sock, file, 40\*sizeof(char), 0), 4, server, server\_);

/\* ожидаем ответа от сервера с резульатами поиска \*/

while ((recv(sock, str, sizeof(str), 0))<0)

cout<<"Сервер не отвечает. Ждём ответ\n";

cout<<"Получен ответ от сервера\n";

cout << str << "\n"; /\* распечатываем содержимое искомого файла \*/

close (sock); /\* закрытие соединения \*/

exit (0);

}

/\*\*

\* функция сервера

\*

\*@ param number номер сервера

\*@ param serv\_1 сокет для первого сервера

\*@ param serv\_2 сокет для второго сервера

\*/

void server\_(int number, struct sockaddr &serv\_1, struct sockaddr &serv\_2)

{

int sock\_serv, sock\_work, pr, fd, n, st;

char file[9]=" "; /\* буфер для хранения имени файла \*/

char er[20];

strcmp(er, "такого файла нет!!");

/\* создание сокета для связи, проверка корретности вызова socket \*/

proverka((sock\_serv = socket(AF\_UNIX,SOCK\_STREAM,0)), 1, serv\_1, serv\_2);

/\* связывание сетевого адреса с идентификатором сокета, проверка вызова bind \*/

proverka((bind (sock\_serv, &serv\_1, sizeof(struct sockaddr))), 6, serv\_1, serv\_2);

proverka(listen (sock\_serv, 5), 7, serv\_1, serv\_2); /\* включение приема соединений \*/

for(;;)

{ /\* цикл организации клиент-серверного взаимодействия \*/

/\* создаем дополнительный сокет для работы с соединением \*/

if ((sock\_work = accept (sock\_serv, NULL,NULL)) == -1)

{ /\* ожидаем установки соединения \*/

cout << "Ошибка соединения\n";

continue;

}

/\* порождаем потомка для обработки сообщений \*/

proverka((pr = fork()), 2, serv\_1, serv\_2);

if (pr == 0)

{

/\* сообщение от клиента с именем файла \*/

proverka (recv(sock\_work, file, sizeof(file), 0), 5, serv\_1, serv\_2);

n = poisk(number, sock\_work, file, serv\_1, serv\_2); /\* поиск файла на сервере \*/

if (n)

{ /\* ести нашли, то сообщаем, на каком сервере найден файл \*/

cout << "Файл на сервере № " << number << "\n";

exit(0);

}

else

{ /\* если файла нет на 1 сервере, то устанавливаем клиент-серверную связь между 1 и 2 серверами \*/

if (number == 1) client\_(2, file, serv\_2, serv\_1);

/\* если файла нет на обоих серверах, отсылаем сообщение об ошибке \*/

else proverka(send (sock\_work, er, sizeof(er), 0), 4, serv\_1, serv\_2);

}

}

break;

}

close(sock\_work); /\* закрываем соединение \*/

wait(&st); /\* ожидание завершения потомка \*/

close(sock\_serv);

exit(0); /\* успешное завершение \*/

}

/\* главная подпрограмма \*/

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

{

int pr1, pr2, pr3, st;

/\* создание и инициализация обобщенных сокетов \*/

struct sockaddr server\_1 = {AF\_UNIX,"serve\_1"};

struct sockaddr server\_2 = {AF\_UNIX,"serve\_2"};

if (argc != 2)

{ /\* при отсутствии входных данных выдаем сообщение об ошибке и завершаемся\*/

perror ("Введите входные данные!!\n");

/\* удаляем из каталога точки входа для файлов, чтобы эти

имена были доступны для дальнейшей работы с ними \*/

unlink (server\_1.sa\_data);

unlink (server\_2.sa\_data);

exit(1);

}

/\* порождаем 3 потомка для работы: 2 сервера и 1 клиент,

проверка корректности fork \*/

proverka(pr1 = fork(), 2, server\_1, server\_2);

if (pr1 == 0) server\_(1, server\_1, server\_2); /\* первый сервер \*/

proverka(pr2 = fork(), 2, server\_1, server\_2);

if (pr2 == 0) server\_(2, server\_2, server\_1); /\* второй сервер \*/

proverka(pr3 = fork(), 2, server\_1, server\_2);

if (pr3 == 0) client\_(1, argv[1], server\_1, server\_2); /\* клиент \*/

/\* ожидаем завершения потомков \*/

wait(&st);

wait(&st);

wait(&st);

/\* освобождение имен файлов для дальнейшей работы с ними \*/

unlink(server\_1.sa\_data);

unlink(server\_2.sa\_data);

return 0; /\* успешное завершение программы \*/

}

**Примеры выполнения**

**Тест 1:**

**Запуск программы:**

./labka8

**Входные данные:**

отсутствуют

**Результат выполнения:**

Введите входные данные!!

**Тест 2:**

**Запуск программы:**

./labka8

**Входные данные:**

1.txt

**Результат выполнения:**

Файл на сервере № 1

Получен ответ от сервера

oiuytrewq

**Тест 3:**

**Запуск программы:**

./labka8

**Входные данные:**

3.txt

**Результат выполнения:**

Сервер не отвечает. Ждём ответ

Файл на сервере № 2

Получен ответ от сервера

dvfbvdfbf

**Тест 4:**

**Запуск программы:**

./labka8

**Входные данные:**

5.txt

**Результат выполнения:**

Сервер не отвечает. Ждём ответ

Получен ответ от сервера

Такого файла нет!!

**Тест 5:**

**Запуск программы:**

./labka8

**Входные данные:**

2.txt

**Результат выполнения:**

Получен ответ от сервера

Файл на сервере № 1

oiuyhjk,m

**Тест 6:**

**Запуск программы:**

./labka8

**Входные данные:**

4.txt

**Результат выполнения:**

Ошибка подключения к серверу

Ошибка подключения к серверу

Ошибка подключения к серверу

Получен ответ от сервера

Файл на сервере № 2

qwertyujk

**Тест 7:**

**Запуск программы:**

./labka8

**Входные данные:**

a.txt

**Результат выполнения:**

ошибка SOKET

**Тест 8:**

**Запуск программы:**

./labka8

**Входные данные:**

1.txt

**Результат выполнения:**

ошибка FORK

**Тест 9:**

**Запуск программы:**

./labka8

**Входные данные:**

1.txt

**Результат выполнения:**

ошибка CONNECT

**Тест 10:**

**Запуск программы:**

./labka8

**Входные данные:**

ss.txt

**Результат выполнения:**

ошибка SEND

**Тест 11:**

**Запуск программы:**

./labka8

**Входные данные:**

44.txt

**Результат выполнения:**

ошибка RECV

**Тест 12:**

**Запуск программы:**

./labka8

**Входные данные:**

44.txt

**Результат выполнения:**

ошибка BIND

**Тест 13:**

**Запуск программы:**

./labka8

**Входные данные:**

44.txt

**Результат выполнения:**

ошибка LISTEN

**Тест 14:**

**Запуск программы:**

./labka8

**Входные данные:**

44.txt

**Результат выполнения:**

ошибка READ

**Тест 15:**

**Запуск программы:**

./labka8

**Входные данные:**

44.txt

**Результат выполнения:**

ошибка OPEN