Министерство образования и науки Российской Федерации

Новосибирский государственный технический университет

Кафедра программных систем и баз данных

Лабораторная работа №8 по дисциплине

«Управление ресурсами в вычислительных системах»

Факультет: ПМИ

Группа: ПМ-04

Студентки: Анисимова К.Н.

Сулейманова К.А.

Вариант: 4

Преподаватель: Быханов К.В.

Куликов И.М.

Новосибирск

2013

1. ***Цель работы***

Практическое освоение механизма сокетов. Построение TCP-соединений для межпроцессного взаимодействия программ клиента и сервера в модели "клиент-сервер".

1. ***Задание***

Организовать взаимодействие типа клиент-сервер. Клиент отсылает строку серверу. Сервер отсылает данную строку на другие сервера, список которых хранится в файле, а там уже осуществляется поиск файлов содержащих данную строку. Результаты поиска отсылаются клиенту.

1. ***Решение***

*Алгоритм работы*

Существует три разных программы – клиент, сервер и основной сервер. Клиент посылает запрос основному серверу. Основной сервер принимает запрос и пересылает его всем серверам из своего списка, собирает ответы и отсылает их клиенту с пометкой от какого сервера был получен ответ.

1. ***Средства программирования***

В данной работе были использованы *структуры:*

* обобщенный сокет (generic socket), определяется в файле <sys/socket.h>:

*struct sockaddr*

*{*

*u\_char sa\_family; /\* Семейство адресов (домен) \*/*

*char sa\_data[]; /\* Адрес сокета \*/*

*};*

* сокеты для связи через сеть, определяются в файле <netinet/in.h>

*struct sockaddr\_in*

*{*

*u\_char sin\_len; /\* Длина поля sockaddr\_in (для FreeBSD) \*/*

*u\_char sin\_family; /\* Семейство адресов (домен) \*/*

*u\_short sin\_port; /\* Номер порта \*/*

*struct in\_addr sin\_addr; /\* IP-адрес \*/*

*char sin\_zero[8]; /\* Поле выравнивания \*/*

*};*

где

*struct in\_addr { n\_int32\_t s\_addr };*

В данной работе были использованы *функции:*

* системный вызов *socket()*

*#include <sys/socket.h>*

*int socket (int domain, int type, int protocol);*

* параметр domain - домен связи, в котором будет использоваться сокет: AF\_INET - для доменаInternet (соединение через сеть),

AF\_UNIX - домен, если процессы находятся на одном и том же компьютере;

* параметр type определяет тип создаваемого сокета:

SOCK\_STREAM - для режима соединений,

SOCK\_DGRAM - для режима дейтаграмм);

* Параметр protocol определяет используемый протокол:

для сокета типа SOCK\_STREAM будет использовать протокол TCP,

для сокета типа SOCK\_DGRAM - протокол UDP.

* системный вызов *bind()* связывает сетевой адрес компьютера с идентификатором сокета.

*#include <sys/types.h>*

*#include <sys/socket.h>*

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

* sockfd - дескриптор файла сокета, созданным с помощью вызова socket();
* address - указатель на обобщённую структуру адреса сокета, к которой преобразуется структура sockaddr\_inв случае передачи данных через сеть;
* size\_t add\_len - размер указанной структуры адреса сокета.
* системный вызов *listen()* (сервер включает прием соединений):

*#include <sys/socket.h>*

*int listen (int sockfd, int queue\_size);*

* sockfd - дескриптор файла сокета, созданный с помощью вызова socket();
* queue\_size - число запросов на соединение с сервером, которые могут стоять в очереди.
* системный вызов *accept()*, принимающий очередное соединение:

*#include <sys/types.h>*

*#include <sys/socket.h>*

*int accept (int sockfd, struct sockaddr \*address, size\_t \*add\_len);*

* sockfd - дескриптор сокета, для которого ведется прием соединений;
* address - указатель на обобщенную структуру адреса сокета с информацией о клиенте; так как связь использует соединение адрес клиента знать не обязательно и допустимо задавать параметр address значением NULL;
* add\_len - размер структуры адреса, заданной параметром address, если значение address не равно NULL.
* для выполнения запроса на подключение к серверному процессу клиент использует системный вызов *connect()*:

*#include <sys/types.h>*

*#include <sys/socket.h>*

*int connect (int csockfd, const struct sockaddr \*address, size\_t add\_len);*

* сsockfd - дескриптор файла сокета клиента, созданным с помощью вызова socket();
* address - указателем на обобщенную структуру адреса сокета, к которой преобразуется структураsockaddr\_in, в случае передачи данных через сеть;
* size\_t add\_len - размер указанной структуры адреса сокета.
* системные вызовы *recv()* и *send()*::

*#include <sys/types.h>*

*#include <sys/socket.h>*

*ssize\_t recv (int sockfd, void \*buffer, size\_t length, int flags);*

*ssize\_t send (int sockfd, const void \*buffer, size\_t length, int flags);*

* socfd - дескриптор сокета, через который читаются или записываются данные;
* buffer - буфер, в который они помещаются или откуда отсылаются через сокет;
* length - размер буфера;
* flags - поле дополнительных опций при получении или передаче данных.

В случае flags=0 вызовы send() и recv() полностью аналогичны системным вызовам read() и write().

1. ***Текст программы***

*client.c*

/\*

\*программа посылает запрос серверу (строку) и получает список

\*файлов, содержащих данную строку

\*/

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <netdb.h>

int main(int argc, char \*argv)

{

int sock;

struct sockaddr\_in addr;

char buf[8192];

//искомая строка

char message[25] = "fyfyfyfyfyfy\0";

//создание сокета

sock = socket(AF\_INET, SOCK\_STREAM, 0);

//проверка

if(sock < 0)

{

perror("socket");

exit(1);

}

//данные сокета связи

addr.sin\_family = AF\_INET;

addr.sin\_port = htons(2525);

addr.sin\_addr.s\_addr = htonl(INADDR\_LOOPBACK);

//коннектимся к серверу

if(connect(sock, (struct sockaddr \*)&addr, sizeof(addr)) < 0)

{

perror("connect");

exit(2);

}

printf("Send %s\n", message);

//отправляем сообщение серверу

int send\_data = send(sock, message, strlen(message)+1, 0);

printf("Sendet %d\n", send\_data);

sleep(8);

//получаем сообщение от сервера

int get\_str = recv(sock, buf, sizeof(buf), 0);

if(get\_str > 0)

printf("Receaved %s\n", buf);

else

printf("Can't recv\n");

close(sock);

return 0;

}

*main\_server.c*

/\*

\*программа принимает от клиента строку и рассылает ее

\*всем серверам из списка, получает от них ответы

\*и отправляет обратно клиенту

\*/

#include <stdio.h>

#include <stdlib.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <netdb.h>

#include <signal.h>

#include <malloc.h>

//размер буфера для приема/отправки сообщений

#define buf\_size 1024

//читаем список серверов

int read(struct sockaddr\_in\* serverlist, int N)

{

int i, port;

char s[20];

char c;

FILE \*f;

f = fopen("serverlist.txt", "r");

for(i = 0; i<N; i++)

{

//считывание данных

fscanf(f, "%s", &s);

fscanf(f, "%d", &port);

printf("%s\t%d\n", s, port);

//если неправильный порт

if(port < 0)

{

printf("error port");

return 0;

}

//получаем IP сервера

struct hostent \* record = gethostbyname(s);

//если неправильный IP

if(record==NULL)

{

printf("error, wrong addres server\n");

return 0;

}

//IP-адрес

struct in\_addr \* addressptr = (struct in\_addr \*) record->h\_addr;

//данные сокета связи

serverlist[i].sin\_family = AF\_INET;

serverlist[i].sin\_addr = \* addressptr;

serverlist[i].sin\_port = htons(port);

}

}

int main()

{

int N = 3, i, bytes\_read;

struct sockaddr\_in serverlist[N];

char buf[buf\_size];

char mes[1024];

//читаем список серверов

read(serverlist, N);

int main\_socket;

struct sockaddr\_in addr; //структура для адреса сокета

//подключение к клиенту

//создаем сокет, если ошибка - выводим

main\_socket = socket(AF\_INET, SOCK\_STREAM, 0);

if(main\_socket < 0)

{

printf("error, socket is not create\n");

exit(1);

}

//стуктура для адреса подключения сокета

addr.sin\_family = AF\_INET;

addr.sin\_port = htons(2525);

addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

//связываем сокет и информацию о сервере

int r1 = bind(main\_socket, (struct sockaddr \*) &addr, sizeof(addr) );

if( r1<0 )

{

printf("error, port is using now\n");

close(main\_socket);

exit(1);

}

//начинаем прослушивать порт

int r2 = listen(main\_socket, 3 );

if( r2 < 0 )

{

printf("error, not access with port \n");

close(main\_socket);

exit(1);

}

while(1)

{

//принимаем соединение

int session\_socket = accept( main\_socket, NULL, NULL);

if ( session\_socket<0 )

{

printf("conection error\n");

exit(1);

}

else

printf("Connection succesfully\n");

bytes\_read = 0;

//пока получаем данные от серверов

while(bytes\_read < 1)

bytes\_read = recv(session\_socket, buf, 1024, 0);

printf("Bites read %d\n", bytes\_read);

printf("We have:\t%s\n",buf);

//проходим по всем серверам

for(i = 0; i< N; i++)

{

int sock1;

//создаем сокет, если ошибка - выводим

sock1 = socket(AF\_INET, SOCK\_STREAM, 0);

if(sock1 < 0)

{

printf("error, socket is not create\n");

exit(1);

}

//пробуем подключиться

int r = connect(sock1, (struct sockaddr \*) &serverlist[i], sizeof(serverlist[i]));

if( r < 0)

{

printf("error, not connect\n");

exit(1);

}

printf("Send to %d\n", i);

//отправляем строку серверу

send(sock1, buf, sizeof(buf), 0);

printf("Sendet to %d\n", i);

sleep(1);

char tmp\_str[1024];

//принимаем имена файлов от серверов

while(1)

{

strcmp(tmp\_str, "");

//принимаем

int res = recv(sock1, tmp\_str, 1024, 0);

//если приняли

if(res > 0)

{

printf("Receved %s from %d\n", tmp\_str, i);

strcat(mes, "\n");

//"склеиваем" результаты

strcat(mes, tmp\_str);

char\* tmp1\_char = strstr(tmp\_str, "\_\_end");

//если всё - заканчиваем принимать результаты

if(tmp1\_char != 0)

{

printf("End of receverd\n");

break;

}

}

};

}

printf("Sending %s\n", mes);

//отправляем результат клиенту

send(session\_socket, mes, strlen(mes), 0);

printf("Sending end\n");

strcpy(mes, ""); //чистим mes для след. запуска

}

}

*server.c*

/\*

\*программа принимает от клиента строку, осуществляет поиск

\*файлов, содержащих данную строку

\*и отправляет обратно клиенту имена файлов

\*/

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <signal.h>

#include <errno.h>

#include <fcntl.h>

#include <sys/wait.h>

//размер буфера дял отправки/приема сообщений

#define buf\_size 1024

//поиск файлов, содержащих определенную строку

/\*fdw дескриптор временного файла

\*str строка для поиска\*/

void search( int fdw, const char\* str )

{

//подменяем стандартный вывод файлом

close(1);

fcntl( fdw, F\_DUPFD, 1 );

close(fdw);

//запускаем скрипт для поиска

if( execlp( "sh", "sh", "search.sh", str, (char\*)0 ) <0 )

exit(1);

exit(0);

}

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

{

char temp\_file\_name[20] = "tmp.txt";

char buf[1024];

//должен быть указан номер порта

if ( argc!=2 )

{

printf("wrong args\n");

exit(1);

}

int port\_number = atol(argv[1]);

if( port\_number<0 )

{

printf("error wrong port\_number\n");

exit(1);

}

//создаем сокет

int main\_socket = socket( AF\_INET, SOCK\_STREAM, 0 );

if( main\_socket<0 )

{

printf("error socket not create\n");

exit(1);

}

//создаем структуру для описания сервера

struct sockaddr\_in server\_info;

server\_info.sin\_family = AF\_INET;

server\_info.sin\_addr.s\_addr = htonl(INADDR\_ANY);

server\_info.sin\_port = htons(port\_number);

//связываем сокет с сервером

int r1 = bind( main\_socket, (struct sockaddr \*) &server\_info, sizeof(server\_info) );

if( r1<0 )

{

printf("port is using now\n");

exit(1);

close(main\_socket);

exit(1);

}

//начинаем прослушивать порт

int r2 = listen(main\_socket, 3);

if( r2<0 )

{

printf("error acsess denaid\n");

exit(1);

close(main\_socket);

exit(1);

}

//номер подключения

int session\_number=0;

//цикл обработки сообщений

while(1)

{

//описание клиента

struct sockaddr\_in client\_info;

unsigned int client\_info\_size = sizeof(client\_info);

//принимаем подключение

int session\_socket = accept( main\_socket, (struct sockaddr \*)&client\_info, &client\_info\_size );

if ( session\_socket<0 )

{

printf("error\n");

exit(1);

}

printf("Receve start\n");

//принимаем строку от основного сервера

recv(session\_socket, buf, sizeof(buf), 0);

printf("Receve done\n");

//создаем временный файл

int tempfdw = open(temp\_file\_name, O\_CREAT | O\_TRUNC | O\_WRONLY, 0644);

if(tempfdw < 0)

{

printf("error, temp\_file not create\n");

close(session\_socket);

exit(1);

}

printf("Search begin\t%s\n", buf);

//создаём параллельный поток для поиска файлов на сервере

int search\_fork = fork();

switch(search\_fork)

{

case 0:

{

//tempfdw - дескриптор временного файла

//buf - искомая строка

search(tempfdw,buf); //ищем

}break;

case -1:

{

printf("NOOOOOOOOOOOOOOOOOOOOOOOOOOOO!\n");

}break;

}

sleep(1);

printf("Search end\n");

close(tempfdw);

//открываем временный файл

FILE \*tempfdr = fopen(temp\_file\_name, "r");

//если не открылся

if(tempfdr == NULL)

{

printf("error, temp\_file not open\n");

close(session\_socket);

exit(1);

}

//считываем из времнного файла сообщения и отправляем серверу

while(fgets(buf, buf\_size, tempfdr))

{

printf("Start send: %s\n", buf);

send(session\_socket, buf, strlen(buf), 0);

printf("Send done\n");

}

//закончилась отправка

char end\_s[] = "\_\_end";

send(session\_socket, end\_s, strlen(end\_s), 0);

close(session\_socket);

}

return 0;

}

1. ***Тесты***

|  |  |
| --- | --- |
| 1. 2 сервера, искомая строка в файлах:   ./server1/file5.txt, ./server2/file2.txt, ./server2/file6.txt  -bash-3.2$ ./client  Send fyfyfyfyfyfy  Sendet 13  file5.txt  \_\_end.  \_\_end.  file2.txt  file6.txt  \_\_end. | 1. 3 сервера, искомая строка в файлах:   ./server1/file5.txt, ./server2/file2.txt, ./server2/file6.txt, ./server3/file7.txt  -bash-3.2$ ./client  Send fyfyfyfyfyfy  Sendet 13  file5.txt  \_\_end  file2.txt  file6.txt  \_\_end.  file7.txt  \_\_end.  \_\_end. |