Лабораторная работа № 2

Организация распределённых вычислений с использованием сокет TCP/IP средствами LinuxAPI.

1. Теоретические сведения

Существует две модели взаимодействия между процессами в сети: модель соединений с протоколом TCP (Transmission Control Protocol), и модель дейтаграмм с протоколом UDP (User Datagram Protocol). Ниже приводятся основные шаги и необходимые системные вызовы для выполнения основных этапов при работе с сокетами в режиме TCP-соединения.

Адресация и создание сокета

Совокупная информация об адресе, порте программы-адресата (абонента), модели соединения, протоколе взаимодействия составляет т.н. сокет (конечная абонентская точка), формально представляющий собой структуру данных. Существует несколько видов сокетов:

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, определяется 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}.

Создается сокет при помощи системного вызова socket().

```
#include <sys/socket.h>
int socket (int domain, int type, int protocol);
```

1. Параметр **domain** - домен связи, в котором будет использоваться сокет (значение **AF_INET** - для домена Internet (соединение через сеть), **AF_UNIX** - домен, если процессы находятся на одном и том же компьютере);

2. Параметр **type** определяет тип создаваемого сокета (значение **SOCK_STREAM** - для режима соединений, **SOCK_DGRAM** - для режима дейтаграмм);

3. Параметр **protocol** определяет используемый протокол (в случае **protocol** = 0 по умолчанию для сокета типа **SOCK_STREAM** будет использовать протокол **TCP**, а сокета типа **SOCK_DGRAM** - протокол **UDP**).

При программировании TCP-соединения должны быть созданы сокеты (системный вызов socket()) и в программе сервера и в программе клиента, при этом в обеих программах сокеты связываются с адресом машины, на которую будет установлена программа сервера. Но, если в программе сервера для определения IP-адреса в структуре сокета может быть использована переменная INADDR_ANY, то в программе клиента для занесения в структуру сокета IP-адреса машины сервера необходимо использовать системный вызов inet_addr().

Сетевые вызовы inet_addr() и inet_ntoa() выполняют преобразования IP-адреса из формата текстовой строки "x.y.z.t" в структуру типа in_addr и обратно.

```
#include <arpa/inet.h>
in_addr_t inet_addr (const char *ip_address);
char * inet_ntoa(const struct in_addr in);
```

Для того чтобы процесс мог ссылаться на адрес своего компьютера, файле <netinet.h> определена переменная INADDR_ANY, содержащая локальный адрес компьютера в формате in_addr_t.

Связывание

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 - указателем на обобщенную структуру адреса сокета, к которой преобразуется структура sockaddr_in, в случае передачи данных через сеть.

size_t add_len - размер указанной структуры адреса сокета.

В случае успешного завершения вызова bind() он возвращает значение 0. В случае ошибки, например, если сокет для этого адреса уже существует, вызов bind() возвращает значение -1. Переменная errno будет иметь при этом значение EADDRINUSE. Операция связывания выполняется только в программе сервера.

Включение приема TCP-соединений

После выполнения связывания с адресом и перед тем, как какой-либо клиент сможет подключиться к созданному сокету, сервер должен включить прием соединений посредством системного вызова listen().

```
#include <sys/socket.h>
int listen (int sockfd, int queue_size);
```

1. **sockfd** - дескриптор файла сокета, созданным с помощью вызова socket(),
2. **queue_size** - число запросов на соединение с сервером, которые могут стоять в очереди.

Данная операция выполняется только в программе сервера.

Прием запроса на установку TCP-соединения

Когда сервер получает от клиента запрос на соединение, он создает новый сокет для работы с новым соединением. Первый же сокет используется только для установки соединения. Дополнительный сокет для работы с соединением создается при помощи вызова accept(), принимающего очередное соединение.

```
#include <sys/types.h>
 #include <sys/socket.h>
int accept (int sockfd, struct sockaddr *address, size_t *add_len);
```

1. **sockfd** - дескриптор сокета, для которого ведется прием соединений;
2. **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.

Возвращаемое значение соответствует идентификатору нового сокета, который будет использоваться для связи. До тех пор, пока от клиента не поступил запрос на соединение, процесс, выдавший системный вызов accept() переводится в состояние ожидания.

Данная операция выполняется только в программе сервера.

Подключение клиента

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);

1. **csockfd** - дескриптор файла сокета клиента, созданным с помощью вызова socket();
2. **address** - указателем на обобщенную структуру адреса сокета, к которой преобразуется структура sockaddr_in, в случае передачи данных через сеть;
3. **size_t add_len** - размер указанной структуры адреса сокета.

В случае успешного завершения вызова connect() он возвращает значение 0. В случае ошибки, системный вызов connect() возвращает значение -1, а переменная errno идентифицирует ошибку. Данная операция выполняется только в программе клиента.

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

Возможные комбинациями констант параметра flags системнного вызова send():

1. MSG_PEEK Процесс может просматривать данные, не "получая" их;
2. MSG_OOB Обычные данные пропускаются. Процесс принимает только срочные данные, например, сигнал прерывания;
3. MSG_WAITALL Возврат из вызова recv произойдет только после получения всех данных.

Возможные комбинациями констант параметра flags системного вызова recv():

1. MSG_OOB Передать срочные (out of band) данные;
2. MSG_DONTROUTE При передаче сообщения игнорируются условия маршрутизации протокола более низкого уровня. Обычно это означает, что сообщение посылается по прямому, а не по самому быстрому маршруту (самый быстрый маршрут не обязательно прямой и может зависеть от текущего распределения нагрузки сети).

Данные операции выполняются и в программе сервера и в программе клиента.

Закрытие TCP-соединения

Закрываются сокеты так же, как и обычные дескрипторы файлового ввода/вывода, - при помощи системного вызова close(). Для сокета SOCK_STREAM ядро гарантирует, что все записанные в сокет данные будут переданы принимающему процессу.

Данные операции выполняются и в программе сервера и в программе клиента.

При реализации UDP сокетов функции listen и accept не нужны, т.к. UDP протокол не создает соединений, следовательно прослушивать порт и ожидать подключений не нужно. Для передачи данных используются функции sendto и recvfrom (См. пример ниже).

Пример реализации TCP клиента

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
#include <netinet/in.h>
char message[] = "Hello there!\n";
char buf[sizeof(message)];
int main()
{
	int sock;
	struct sockaddr_in addr;
	sock = socket(AF_INET, SOCK_STREAM, 0);
	if (sock < 0)
	{
		perror("socket");
		exit(1);
	}
	addr.sin_family = AF_INET;
	addr.sin_port = htons(3425); // или любой другой порт...
	addr.sin_addr.s_addr = htonl(INADDR_LOOPBACK);
	if (connect(sock, (struct sockaddr *)&addr, sizeof(addr)) < 0)
	{
		perror("connect");
		exit(2);
	}
	send(sock, message, sizeof(message), 0);
	recv(sock, buf, sizeof(message), 0);

	printf(buf);
	close(sock);
	return 0;
}
```

Пример реализации TCP сервера

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
int main()
{
	int sock, listener;
	struct sockaddr_in addr;
	char buf[1024];
	int bytes_read;
	listener = socket(AF_INET, SOCK_STREAM, 0);
	if (listener < 0)
	{
		perror("socket");
		exit(1);
```

```
        }

        addr.sin_family = AF_INET;
        addr.sin_port = htons(3425);
        addr.sin_addr.s_addr = htonl(INADDR_ANY);
        if (bind(listener, (struct sockaddr *)&addr, sizeof(addr)) < 0)
        {
                perror("bind");
                exit(2);
        }
        listen(listener, 1);

        while (1)
        {
                sock = accept(listener, NULL, NULL);
                if (sock < 0)
                {
                        perror("accept");
                        exit(3);
                }
                while (1)
                {
                        bytes_read = recv(sock, buf, 1024, 0);
                        if (bytes_read <= 0) break;
                        send(sock, buf, bytes_read, 0);
                }

                close(sock);
        }

        return 0;
}
```

Пример реализации UDP клиента

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
char msg1[] = "Hello there!\n";
char msg2[] = "Bye bye!\n";
int main()
{
        int sock;
        struct sockaddr_in addr;
        sock = socket(AF_INET, SOCK_DGRAM, 0);
```

```
	if (sock < 0)
	{
		perror("socket");
		exit(1);
	}
	addr.sin_family = AF_INET;
	addr.sin_port = htons(3425);
	addr.sin_addr.s_addr = htonl(INADDR_LOOPBACK);
	sendto(sock, msg1, sizeof(msg1), 0,
		(struct sockaddr *)&addr, sizeof(addr));
	connect(sock, (struct sockaddr *)&addr, sizeof(addr));
	send(sock, msg2, sizeof(msg2), 0);
	close(sock);
	return 0;
}
```

Пример реализации UDP сервера

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
int main()
{
	int sock;
	struct sockaddr_in addr;
	char buf[1024];
	int bytes_read;
	sock = socket(AF_INET, SOCK_DGRAM, 0);
	if (sock < 0)
	{
		perror("socket");
		exit(1);
	}

	addr.sin_family = AF_INET;
	addr.sin_port = htons(3425);
	addr.sin_addr.s_addr = htonl(INADDR_ANY);
	if (bind(sock, (struct sockaddr *)&addr, sizeof(addr)) < 0)
	{
		perror("bind");
		exit(2);
	}
	while (1)
	{
```

```
                bytes_read = recvfrom(sock, buf, 1024, 0, NULL, NULL);
                buf[bytes_read] = '\0';
                printf(buf);
        }

        return 0;
} Низкоуровневые сокеты
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
struct UdpHeader
{
        u_short src_port;
        u_short targ_port;
        u_short length;
        u_short checksum;
};
char message[] = "Hello there!\n";
char msgbuf[1024];
int main()
{
        int sock;
        struct sockaddr_in addr;
        struct UdpHeader header;
        sock = socket(AF_INET, SOCK_RAW, IPPROTO_UDP);
        if (sock < 0)
        {
                perror("socket");
                exit(1);
        }
        addr.sin_family = AF_INET;
        addr.sin_addr.s_addr = htonl(INADDR_LOOPBACK);

        header.targ_port = htons(3425);
        header.length = htons(sizeof(header)+sizeof(message));
        header.checksum = 0;

        memcpy((void *)msgbuf, (void *)&header, sizeof(header));
        memcpy((void *)(msgbuf + sizeof(header)), (void *)message,
                sizeof(message));
        sendto(sock, msgbuf, sizeof(header)+sizeof(message), 0,
                (struct sockaddr *)&addr, sizeof(addr));
        close(sock);
        return 0;
}
```

2. Задание на лабораторную работу.

Задания представленные ниже выполняются с использованием API Linux.

Варианты задания:

Распределенное вычисление площади поверхности фигуры одним из предложенных по варианту методов (в таблице варианты 1-20). Исходными данными для вычисления являются: ● отрезок [A;B]; ● точность вычислений; ● количество клиентов решающих задачу.	4-5 баллов
Распределенное вычисление объема поверхности фигуры одним из предложенных по варианту методов (в таблице варианты 21-32). Исходными данными для вычисления являются: ● отрезок [A;B] и [C;D]; ● точность вычислений; ● количество клиентов решающих задачу.	4-5 баллов
Задания написаны в таблице	6-8 баллов
алгоритм выбрать в соответствии с вариантом задания. Есть один сервер, к которому подключено N клиентов. На сервере указываете матрицу, которая считывается и её части рассылаются клиентам (не полностью !). В результате вы должны получить вектор решений, который сохраняется в текстовый файл. В программе также задается количество клиентов.	9-10 баллов

Обязательные компоненты отчета:

- Краткое описание алгоритма
- Описание передаваемых по сети данных
- Верификация с любым мат. пакетом (mathcad, matlab, octave или scilab)
- Сравнение скорости решения для одного клиента, двух и четырех

Варианты согласуются с преподавателем:

1.	Вычислить площадь фигуры, описанную следующей функцией f(x)= $e^{-x}sin(x)$ методом трапеций.
2	Вычислить площадь фигуры, описанную следующей функцией f(x)= $e^{x}arctg(x)$ методом центральных прямоугольников.
3	Вычислить площадь фигуры, описанную следующей функцией f(x)= $x^{3}cos(x)$ методом левых прямоугольников.
4	Вычислить площадь фигуры, описанную следующей функцией f(x)= $e^{cos(x)}x^{2}$ методом центральных прямоугольников.
5	Вычислить площадь фигуры, описанную следующей функцией f(x)= $x^{2}2^{\sqrt[3]{xcos(x)sin(x)}}$ методом правых прямоугольников.

6	Вычислить площадь фигуры, описанную следующей функцией f(x)=$x^3 sin(x)$ методом Симпсона.
7	Вычислить площадь фигуры, описанную следующей функцией f(x)=$-x^4 cos(x)$ методом Монте-Карло.
8	Вычислить площадь фигуры, описанную следующей функцией f(x)=$\frac{e^{xcos(x)}}{x^2+1}$ методом трапеций.
9	Вычислить площадь фигуры, описанную следующей функцией f(x)=$\frac{2^{xcos(x)sin(x)}}{sin^2 x+cos^2 x}$ методом центральных прямоугольников.
10	Вычислить площадь фигуры, описанную следующей функцией f(x)=$x^3 2^{\sqrt[3]{xsin(x)}}$ методом левых прямоугольников.
11	Вычислить площадь фигуры, описанную следующей функцией f(x)=$x^3 2^{\sqrt[3]{xcos(x)}}$ методом центральных прямоугольников.
12	Вычислить площадь фигуры, описанную следующей функцией f(x)=$x^2 2^{\sqrt[3]{xsin(x)}}$ методом правых прямоугольников.
13	Вычислить площадь фигуры, описанную следующей функцией f(x)=$4^x e^{cos(x)}$ методом Симпсона.
14	Вычислить площадь фигуры, описанную следующей функцией f(x)=$\frac{e^{xsin(x)}}{x^2+1}$ методом трехточечной квадратуры Гаусса-Лежандра.
15	Вычислить площадь фигуры, описанную следующей функцией f(x)=$2^{x\sqrt{e^{sin(x)cos(x)}}}$ методом трапеций.
16	Вычислить площадь фигуры, описанную следующей функцией f(x)=$2^{x\sqrt{e^{sin(x)}}}$ методом центральных прямоугольников.
17	Вычислить интеграл функции f(x)=$4^x sin(x)$ методом левых прямоугольников.
18	Вычислить площадь фигуры, описанную следующей функцией f(x)=$e^{cos(x)} 2^x$ методом центральных прямоугольников.
19	Вычислить площадь фигуры, описанную следующей функцией f(x)= $e^{-cos(x)} 2^{\sqrt[3]{sin(x)}}$ методом правых прямоугольников.
20	Вычислить площадь фигуры, описанную следующей функцией f(x)=$x^3 e^{cos(x)}$ методом Симпсона.
21	Вычислить объем фигуры, описанный следующей функцией f(x)=$\frac{e^{xcos(y)}}{y^2+1}$ методом трапеций.
22	Вычислить объем фигуры, описанный следующей функцией f(x)=$\frac{2^{ycos(x)sin(x)}}{sin^2 y+cos^2 y}$ методом центральных прямоугольников.
23	Вычислить объем фигуры, описанный следующей функцией f(x)=$y^3 2^{\sqrt[3]{ysin(x)}}$ методом левых прямоугольников.

24	Вычислить объем фигуры, описанный следующей функцией f(x)=$x^3 2^{\sqrt[3]{y\cos(y)}}$ методом центральных прямоугольников.
25	Вычислить объем фигуры, описанный следующей функцией f(x)=$x^2 2^{\sqrt[3]{y\sin(x)}}$ методом правых прямоугольников.
26	Вычислить объем фигуры, описанный следующей функцией f(x)=$4^y e^{\cos(x)}$ методом Симпсона.
27	Вычислить объем фигуры, описанный следующей функцией f(x)=$\frac{e^{y\sin(x)}}{x^2+1}$ методом трехточечной квадратуры Гаусса-Лежандра.
28	Вычислить объем фигуры, описанный следующей функцией f(x)=$2^{y\sqrt{e^{\sin(x)\cos(y)}}}$ методом трапеций.
29	Вычислить объем фигуры, описанный следующей функцией f(x)=$4^y \sin(x)$ методом левых прямоугольников.
30	Вычислить объем фигуры, описанный следующей функцией f(x)=$e^{\cos(x)} 2^y$ методом центральных прямоугольников.
31	Вычислить объем фигуры, описанный следующей функцией f(x)=$e^{-\cos(x)} 2^{\sqrt[3]{\sin(y)}}$ методом правых прямоугольников.
32	Вычислить объем фигуры, описанный следующей функцией f(x)=$y^3 e^{\cos(x)} + x^2$ методом Симпсона.
33	Распределенное умножение матриц, распределение по строкам (блоками)
34	Распределенное умножение матриц, распределение по столбцам (блоками)
35	Распределенное умножение матриц, распределение по строкам (циклически)
36	Распределенное умножение матриц, распределение по столбцам (циклически)
37	Распределенное умножение матриц, клеточное распределение
38	Распределенное умножение матриц, через внешнее произведение
39	Распределенная сортировка, для сбора от клиентов использовать mergesort, на клиентах использовать сортировку вставкой
40	Распределенная сортировка, для сбора от клиентов использовать mergesort, на клиентах использовать сортировку mergesort
41	Распределенная сортировка, для сбора от клиентов использовать mergesort, на клиентах использовать сортировку quicksort
42	Распределенная сортировка, для сбора от клиентов использовать mergesort, на клиентах использовать сортировку heapsort
43	Распределенная сортировка, для сбора от клиентов использовать mergesort, на клиентах использовать сортировку treesort (симметричный обход бинарного дерева

44	Распределенная одномерная оптимизация методом дихотомии
45	Распределенная одномерная оптимизация методом градиентного спуска
46	Решение СЛАУ через обратную матрицу, для получения обратной матрицы использовать распределение по столбцам
47	Решение СЛАУ через обратную матрицу, для получения обратной матрицы использовать распределение по строкам
48	Найти обратную матрицу (используя метод Гаусса), распределение по строкам (циклическое)
49	Найти обратную матрицу (используя метод Гаусса), распределение по столбцам (циклическое)
50	Решение СЛАУ методом Гаусса, распределение по столбцам (блоками)
51	Решение СЛАУ методом Гаусса, распределение по строкам (блоками)
52	Решение СЛАУ методом Гаусса, распределение по столбцам (циклическое)
53	Решение СЛАУ методом Гаусса, распределение по строкам (циклическое)
54	Решение СЛАУ методом Гаусса, клеточное распределение
55	Решение СЛАУ методом Жардана-Гауса, распределение по столбцам (циклическое)
56	Решение СЛАУ методом Жардана-Гауса, распределение по строкам(циклическое)
57	Найти определитель матрицы, распределение по строкам (циклическое)
58	Найти определитель матрицы, распределение по столбцам (циклическое)
59	Найти обратную матрицу (через единичную), распределение по строкам (циклическое)
60	Найти обратную матрицу (через единичную), распределение по столбцам (циклическое)

3. Контрольные вопросы

1. Стек протоколов TCP / IP
2. IP - адресация
3. Механизм сокетов
4. Особенности реализации сокетов TCP / IP в Linux
5. Основные функции необходимые для реализации сетевого взаимодействия по протоколу TCP в ОС Linux.
6. Основные функции необходимые для реализации сетевого взаимодействия по протоколу UDP в ОС Linux
7. Отличия протоколов TCP от UDP.

8. Последовательность вызова функций для взаимодействия по протоколу TCP на стороне клиента в ОС Linux
9. Последовательность вызова функций для взаимодействия по протоколу UDP на стороне клиента ОС Linux
10. Последовательность вызова функций для взаимодействия по протоколу TCP на стороне сервера ОС Linux
11. Последовательность вызова функций для взаимодействия по протоколу UDP на стороне сервера ОС Linux
12. Отличия блокирующих от неблокирующих сокетов.