Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Институт «Электронных и информационных систем»

Кафедра «Информационных систем и технологий»

**Семейство протоколов TCP/IP. Сокеты в UNIX и работа с ними**

Лабораторная работа №11 по учебной дисциплине «Операционные системы»

По направлению 09.03.01 «Информатика и вычислительная техника»

Отчёт

Принял преподаватель:

\_\_\_\_\_\_\_\_\_ Ананьев В. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Выполнил студент группы 8091:

\_\_\_\_\_\_\_\_\_ Баранов А. С.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Великий Новгород

2020

**Цель работы:**

В результате выполнения работы, необходимо познакомиться с механизмами работы сокетов в UNIX.

**Задание:**

Написать программу-клиент и программу-сервер для выполнения следующего сценария: клиент генерирует случайным образом массив целых чисел с заданным размером и диапазоном значений; клиент передаёт по сети массив на сервер; сервер сортирует массив по убыванию и возвращает его клиенту; клиент выводит на экран отсортированный массив и время ожидания ответа от сервера.

При запуске программы-сервера через параметры командной строки передаётся порт, на котором будет работать сервер. Если данный порт занят, программа-сервер должна получить порт по усмотрению системы. При успешном запуске сервер должен выдать на экран номер порта, на котором он работает.

При запуске программы-клиента через параметры командной строки передаётся IP-адрес и порт (сокет) программы-сервера в формате <IP-address>:<port> (например, 192.168.0.100:4242). После запуска программы-клиента пользователь вводит размер массива, минимальное и максимальное значение элементов.

**Содержание serv.cpp**

|  |
| --- |
| #include <iostream> |
| #include <sys/types.h> | |
| #include <sys/socket.h> | |
| #include <netinet/in.h> | |
| #include <arpa/inet.h> | |
| #include <string.h> | |
| #include <unistd.h> | |
|  | |
| using namespace std; | |
|  | |
| int compare\_ints(const void \*a, const void \*b) | |
| { | |
| return (\*((int \*)b) - \*((int \*)a)); | |
| } | |
|  | |
| void printmas(int\* arr, int nn) | |
| { | |
| for(int i = 0; i < nn; i++) | |
| { | |
| cout << arr[i] << " "; | |
| } | |
| cout << endl; | |
| } | |
|  | |
| int main(int argc, char\* argv[]) | |
| { | |
| int sockfd, newsockfd; socklen\_t clilen; | |
| int line[1000]; | |
| struct sockaddr\_in servaddr, cliaddr; | |
|  | |
|  | |
|  | |
| sockfd = socket(PF\_INET, SOCK\_STREAM, 0 ); | |
| if (sockfd < 0) | |
| { | |
| perror("socket() error"); | |
| exit(-1); | |
| } | |
|  | |
| bzero(&servaddr, sizeof(servaddr)); | |
| servaddr.sin\_family = PF\_INET; | |
| servaddr.sin\_port = htons(atoi(argv[1])); | |
| servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY); | |
|  | |
|  | |
| if(bind(sockfd, (struct sockaddr\*) &servaddr, sizeof(servaddr)) < 0) | |
| { | |
| servaddr.sin\_port = htons(0); | |
| if(bind(sockfd, (struct sockaddr\*) &servaddr, sizeof(servaddr)) < 0) | |
| { | |
| perror("bind() error"); | |
| close(sockfd); | |
| exit(-1); | |
| } | |
| } | |
| cout << ntohs(servaddr.sin\_port) << " port is used."<< endl; | |
|  | |
|  | |
| if (listen(sockfd, 1) < 0) | |
| { | |
| perror(NULL); | |
| close(sockfd); | |
| exit(-1); | |
| } | |
|  | |
| newsockfd = accept(sockfd, NULL, NULL); | |
| if ( newsockfd < 0) | |
| { | |
| perror("accept() error"); | |
| close(sockfd); | |
| exit(-1); | |
| } | |
| int n; | |
| read(newsockfd, &n , 4); | |
|  | |
| read(newsockfd, line, sizeof(int)\*n); | |
| qsort(line, n,sizeof(int), compare\_ints); | |
| write(newsockfd, line, sizeof(int)\*n); | |
|  | |
| close(sockfd); | |
| close(newsockfd); | |
|  | |
| return 0; | |
|  | } | |

**Содержание cli.cpp**

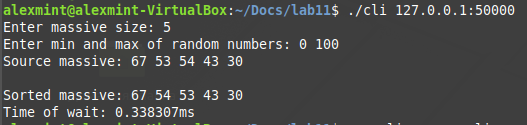
|  |
| --- |
| #include <iostream> |
| #include <sys/types.h> |
| #include <sys/socket.h> |
| #include <netinet/in.h> |
| #include <arpa/inet.h> |
| #include <string.h> |
| #include <unistd.h> |
| #include <errno.h> |
| #include <chrono> |
|  |
| using namespace std; |
| using namespace std::chrono; |
|  |
|  |
|  |
| void randrand(int\* arr, int minn, int maxn, int n) |
| { |
| srand(time(NULL)); |
| int r = maxn - minn + 1; |
| for (int i = 0; i < n; i++) |
| { |
| arr[i] = minn + rand() % r; |
| } |
| } |
|  |
| void printmas(int\* arr, int nn) |
| { |
| for(int i = 0; i < nn; i++) |
| { |
| cout << arr[i] << " "; |
| } |
| cout << endl; |
| } |
|  |
| int ipget(char\* arg, char\* ip) |
| { |
| int iplen; |
| int i = 0; |
| while(arg) |
| { |
| if(arg[i] == ':') |
| { |
| iplen = i; |
| break; |
| }i++; |
| }i = 0; |
| for (char\* l = ip; l < ip + iplen ; l++) |
| { |
| \*l = \*(arg + i); |
| i++; |
| } |
| \*(ip + i) = '\0'; |
|  |
| return iplen; |
| } |
|  |
| int main(int argc, char\* argv[]) |
| { |
| if(argc != 2 ) |
| {cout << "Error: program needs socket." << endl;} |
|  |
| int ipl; char ip[15]; |
|  |
| ipl = ipget(argv[1], ip); |
| char\* prt = new char[strlen(argv[1]) - ipl + 1]; |
|  |
| prt = argv[1] + ipl+1; |
|  |
| //---------------------------------------------------------- |
| int sockfd, len, n, maslen, min, max; |
| struct sockaddr\_in servaddr, cliaddr; |
| cout << "Enter massive size: "; |
| cin >> maslen; |
| int sendline[maslen],recvline[maslen]; |
|  |
| cout << "Enter min and max of random numbers(separated by space): "; |
| cin >> min >> max; |
|  |
| randrand(sendline, min, max, maslen); |
|  |
| cout << "Source massive: "; |
| printmas(sendline, maslen); |
|  |
| sockfd = socket(PF\_INET, SOCK\_STREAM, 0); |
| if (sockfd < 0) |
| { |
| perror("socket() error"); |
| exit(-1); |
| } |
|  |
| //---------------------------------------------------------- |
|  |
| bzero(&servaddr, sizeof(servaddr)); |
| servaddr.sin\_family = PF\_INET; |
| servaddr.sin\_port = htons(atoi(prt)); |
| if (inet\_aton(ip, &servaddr.sin\_addr) == 0) |
| { |
| cout << "invalid IP!" << endl; |
| close(sockfd); |
| exit(-1); |
| } |
|  |
| //------------------------------------------------------------ |
| if(connect(sockfd, (struct sockaddr\*) &servaddr, sizeof(servaddr)) < 0) |
| { |
| perror("Connect() error"); |
| close(sockfd); |
| exit(-1); |
| } |
| //------------------------------------------------------------ |
| steady\_clock::time\_point start,end; |
|  |
|  |
|  |
| if(write(sockfd, (void\*)&maslen, 4)< 0) |
| { |
| perror("write()1 error"); |
| close(sockfd); |
| exit(-1); |
| } |
|  |
| if(write(sockfd, sendline, sizeof(int)\*maslen)< 0) |
| { |
| perror("write()2 error"); |
| close(sockfd); |
| exit(-1); |
| } |
| start = steady\_clock::now(); |
| //------------------------------------------------------------ |
| int r = read(sockfd, recvline, sizeof(int)\*maslen); |
| end = steady\_clock::now(); |
| if (r < 0) |
| { |
| cout << "Error with receiving!" << endl; |
| close(sockfd); |
| exit(-1); |
| } |
|  |
| duration<double, milli> waittime = end - start; |
| cout << endl << "Sorted massive: "; |
| printmas(recvline, maslen); |
| cout << "Time of wait: " << to\_string(waittime.count()) << "ms" << endl; |
|  |
|  |
| close(sockfd); |
| return 0; |
| } |

**Результат выполнения программы:**

Запуск сервера:



Работа клиента:



**Вывод:**

В процессе выполнения данной лабораторной работы, я на практике познакомился с работой сокетов и использовании TCP/IP в UNIX.