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:

\_\_\_\_\_\_\_ Шляханов Д. А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Великий Новгород

2020

**Цель работы:** 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.

Протокол транспортного уровня: UDP.

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 файла server.c:**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <time.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <errno.h>

#include <unistd.h>

int comparator(const void \*a, const void \*b)

{

return (\*((int \*)b) - \*((int \*)a));

}

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

{

int listener, sock;

int len = 1024;

int buff[len];

int bytes\_read;

listener = socket(PF\_INET, SOCK\_STREAM, 0);

struct sockaddr\_in addr, cliaddr;

bzero(&addr, sizeof(addr));

addr.sin\_family = AF\_INET;

addr.sin\_port = htons(atoi(argv[1]));

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

if (bind(listener, (struct sockaddr \*)&addr, sizeof(addr)) < 0)

{

addr.sin\_port = 0;

if (bind(listener, (struct sockaddr \*)&addr, sizeof(addr)) < 0)

{

perror("Bind");

close(listener);

exit(2);

}

}

socklen\_t servlen = sizeof(addr);

getsockname(listener, (struct sockaddr \*)&addr, &servlen);

printf("Listening on port: %d\n", ntohs(addr.sin\_port));

listen(listener, 1);

while(1)

{

socklen\_t clilen = sizeof(cliaddr);

sock = accept(listener, (struct sockaddr \*)&cliaddr, &clilen);

while (1)

{

bytes\_read = recv(sock, buff, len, 0);

if(bytes\_read <= 0) break;

qsort((int \*)buff, bytes\_read \* sizeof(char) / sizeof(int), sizeof(int), comparator);

send(sock, (int \*)buff, bytes\_read \* sizeof(char) / sizeof(int), 0);

}

close(sock);

}

}

**Содержание файла client.c:**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <time.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <errno.h>

#include <unistd.h>

void random\_nums(int \*nums, int n, int min, int max)

{

srand((time(0)));

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

{

nums[i] = min + rand() % (max - min + 1);

}

}

void print\_nums(int \*nums, int c)

{

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

{

printf("%d ", nums[i]);

}

printf("\n");

}

long int time\_server(struct timespec start, struct timespec end)

{

return 1000000000\*(end.tv\_sec - start.tv\_sec)+(end.tv\_nsec - start.tv\_nsec);;

}

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

{

int n, min, max;

int port = atoi(strchr(argv[1], ':') + 1);

int pointer = strchr(argv[1], ':') - argv[1];

char ip[pointer + 1];

strncpy(ip, argv[1], pointer);

ip[pointer] = '\0';

printf("Enter array length, min, max: ");

scanf("%d %d %d", &n, &min, &max);

int arr[n];

int sock;

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

struct sockaddr\_in addr;

bzero(&addr, sizeof(addr));

addr.sin\_family = AF\_INET;

addr.sin\_port = htons(port);

inet\_aton(ip, &addr.sin\_addr);

random\_nums(arr, n, min, max);

print\_nums(arr, n);

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

{

perror("connect");

exit(2);

}

struct timespec start, end;

send(sock, arr, n \* sizeof(int), 0);

clock\_gettime (CLOCK\_REALTIME, &start);

recv(sock, arr, n \* sizeof(int), 0);

clock\_gettime (CLOCK\_REALTIME, &end);

print\_nums(arr, n);

printf("Server time: %ld нс\n", time\_server(start, end));

close(sock);

}

**Вывод:** В процессе выполнения лабораторной работы я на практике познакомился с работой сокетов в UNIX.