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:

\_\_\_\_\_\_\_\_\_ Ефимов 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 server.c**

**#include <sys/types.h>**

**#include <sys/socket.h>**

**#include <netinet/in.h>**

**#include <arpa/inet.h>**

**#include <string.h>**

**#include <stdio.h>**

**#include <errno.h>**

**#include <unistd.h>**

**#include <limits.h>**

**#include <stdlib.h>**

**int compare\_ints(const void \*a, const void \*b)**

**{**

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

**}**

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

**{**

**int sockfd;**

**struct sockaddr\_in servaddr;**

**bzero(&servaddr, sizeof(servaddr));**

**servaddr.sin\_family = AF\_INET;**

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

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

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

**if (bind(sockfd, (struct sockaddr \*)&servaddr, sizeof(servaddr)) < 0)**

**{**

**servaddr.sin\_port = 0;**

**if (bind(sockfd, (struct sockaddr \*)&servaddr, sizeof(servaddr)) < 0)**

**{**

**perror(NULL);**

**close(sockfd);**

**exit(1);**

**}**

**}**

**socklen\_t servlen = sizeof(servaddr);**

**listen(sockfd, 5);**

**getsockname(sockfd, (struct sockaddr \*)&servaddr, &servlen);**

**printf("port: %d\n", ntohs(servaddr.sin\_port));**

**while (1)**

**{**

**struct sockaddr\_in cliaddr;**

**socklen\_t clilen = sizeof(cliaddr);**

**int newsockfd = accept(sockfd, (struct sockaddr \*)&cliaddr, &clilen);**

**int len;**

**read(newsockfd, &len, sizeof(int));**

**int line[len];**

**read(newsockfd, line, len \* sizeof(int));**

**qsort(line, len, sizeof(int), compare\_ints);**

**write(newsockfd, line, len \* sizeof(int));**

**}**

**}**

**Содержание cilent.c**

**#include <sys/types.h>**

**#include <sys/socket.h>**

**#include <netinet/in.h>**

**#include <arpa/inet.h>**

**#include <sys/time.h>**

**#include <string.h>**

**#include <stdio.h>**

**#include <errno.h>**

**#include <unistd.h>**

**#include <stdlib.h>**

**#include <time.h>**

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

**{**

**srand((unsigned)(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 timedifference(struct timeval t0, struct timeval t1)**

**{**

**return (t1.tv\_sec - t0.tv\_sec) \* 1000000 + (t1.tv\_usec - t0.tv\_usec);**

**}**

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

**{**

**int n = 0, min, max;**

**while (n <= 0)**

**{**

**printf("amount of numbers: ");**

**scanf("%d", &n);**

**}**

**printf("min: ");**

**scanf("%d", &min);**

**printf("max: ");**

**scanf("%d", &max);**

**int sockfd;**

**char sendline[n \* sizeof(int)], recvline[n \* sizeof(int)];**

**struct sockaddr\_in servaddr;**

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

**char \*sep = strchr(argv[1], ':');**

**bzero(&servaddr, sizeof(servaddr));**

**servaddr.sin\_family = AF\_INET;**

**servaddr.sin\_port = htons(atoi(sep + 1));**

**sep[0] = 0;**

**inet\_aton(argv[1], &servaddr.sin\_addr);**

**connect(sockfd, (struct sockaddr \*)&servaddr, sizeof(servaddr));**

**fill\_random\_nums((int \*)sendline, n, min, max);**

**print\_nums((int \*)sendline, n);**

**struct timeval start, end;**

**write(sockfd, &n, sizeof(int));**

**write(sockfd, sendline, sizeof(sendline));**

**gettimeofday(&start, 0);**

**int n1 = read(sockfd, recvline, sizeof(recvline));**

**gettimeofday(&end, 0);**

**print\_nums((int \*)recvline, n);**

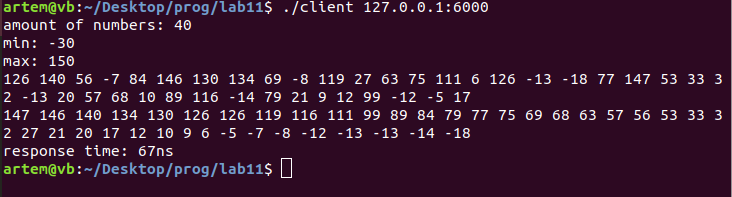
**printf("response time: %ldns\n", timedifference(start, end));**

**close(sockfd);**

**}**

**Результат выполнения программы:**





**Вывод:**

В процессе выполнения данной лабораторной работы, я на практике познакомился с работой сокетов в UNIX.