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»

**Клиент-сервер. Создание Web-серверов.**

Лабораторная работа №12 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 программы, объединить полученные в процессе выполнения предыдущих лабораторных работ знания в рамках одной комплексной программы.

**Задание:**

Написать программу-клиент и программу-сервер для выполнения следующей задачи: сервер запускается и ждёт запросов от клиентов; клиент подключается к серверу и может запрашивать файлы, находящиеся на сервере (в папке с программой-сервером), задавая полное имя файла; если файл найден, сервер передаёт файл клиенту, если не найден, то сервер возвращает сообщение о том, что файл не найден.

Сервер должен параллельно обслуживать несколько клиентов (для каждого свой процесс). Клиент может запрашивать файлы последовательно вводя имена. Выход из программы клиента можно осуществить по текстовой команде (например, «exit»). Сервер так же должен иметь возможность вводить команды (например, для закрытия программы-сервера).

Протокол транспортного уровня – TCP.

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).

**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>**

**#define MAX\_FILENAME\_SIZE 256**

**long fsize(FILE \*fp)**

**{**

**fseek(fp, 0L, SEEK\_END);**

**long sz = ftell(fp);**

**fseek(fp, 0, SEEK\_SET);**

**return sz;**

**}**

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

**if (fork() == 0)**

**{**

**while (1)**

**{**

**struct sockaddr\_in cliaddr;**

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

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

**if (fork() == 0)**

**continue;**

**while (1)**

**{**

**char filename[MAX\_FILENAME\_SIZE];**

**int n = read(newsockfd, filename, MAX\_FILENAME\_SIZE);**

**if (n == 0)**

**{**

**close(newsockfd);**

**exit(0);**

**}**

**FILE \*fin = fopen(filename, "r");**

**if (fin == NULL)**

**{**

**long statusmsg = -1;**

**write(newsockfd, &statusmsg, sizeof(statusmsg));**

**}**

**else**

**{**

**long filesize = fsize(fin);**

**write(newsockfd, &filesize, sizeof(filesize));**

**char msg[filesize];**

**fread(msg, sizeof(char), filesize, fin);**

**fclose(fin);**

**write(newsockfd, msg, sizeof(msg));**

**}**

**}**

**}**

**}**

**else**

**{**

**printf("ready to recive commands\n");**

**char command[MAX\_FILENAME\_SIZE];**

**while (1)**

**{**

**scanf("%s", command);**

**if (strcmp(command, "exit") == 0)**

**{**

**exit(0);**

**}**

**else if (strcmp(command, "help") == 0)**

**{**

**printf("available commands:\n");**

**printf("exit - close programm\n");**

**printf("help - show available commands\n");**

**}**

**else**

**{**

**printf("unknown command, use help to see available commands\n");**

**}**

**}**

**}**

**}**

**Содержание client.c**

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

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

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

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

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

**#include <fcntl.h>**

**#include <string.h>**

**#include <stdio.h>**

**#include <errno.h>**

**#include <unistd.h>**

**#include <limits.h>**

**#include <stdlib.h>**

**#define MAX\_FILENAME\_SIZE 256**

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

**{**

**struct sockaddr\_in servaddr;**

**int 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);**

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

**{**

**printf("cant connect to server\n");**

**return 1;**

**}**

**printf("ready to recive commands\n");**

**char command[MAX\_FILENAME\_SIZE];**

**while (1)**

**{**

**scanf("%s", command);**

**if (strcmp(command, "exit") == 0)**

**{**

**exit(0);**

**}**

**else if (strcmp(command, "помощь") == 0)**

**{**

**printf("available commands:\n");**

**printf("exit - close programm\n");**

**printf("help - show available commands\n");**

**printf("other command will be a filename\n");**

**}**

**else**

**{**

**write(sockfd, command, strlen(command) + 1);**

**long status;**

**int n = read(sockfd, &status, sizeof(status));**

**if (n == 0)**

**{**

**close(sockfd);**

**printf("connection is lost\n");**

**exit(1);**

**}**

**else if (status == -1)**

**{**

**printf("file not found on server\n");**

**}**

**else**

**{**

**printf("downloading file...\n");**

**long filesize = status;**

**char output[strlen(command) + strlen("d\_") + 1];**

**strcpy(output, "d\_");**

**strcat(output, command);**

**printf("%s\n", output);**

**int output\_file = open(output, O\_RDWR | O\_CREAT, 0600);**

**ftruncate(output\_file, filesize);**

**char \*output\_file\_data = (char \*)mmap(NULL, filesize, PROT\_WRITE | PROT\_READ, MAP\_SHARED, output\_file, 0);**

**read(sockfd, output\_file\_data, filesize);**

**munmap(output\_file\_data, filesize);**

**close(output\_file);**

**printf("downloading finished\n");**

**}**

**}**

**}**

**close(sockfd);**

**}**

**Вывод:**

В результате выполнения данной лабораторной работы, я на практике закрепил знания о различных аспектах UNIX.