Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования "Новгородский государственный университет имени Ярослава Мудрого"

Кафедра «Информационных технологий и систем»

Дисциплина «Операционные системы»

Отчет по лабораторной работе

«Клиент-сервер. Создание Web-серверов.»

Выполнил студент группы 9091

\_\_\_\_\_\_\_\_\_\_\_\_\_/Малинин Никита Валерьевич/

Подпись ФИО

Принял преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_/Ананьев Владислав Валерьевич/

Подпись ФИО

Великий Новгород

2021

**Цель лабораторной работы**

Цель работы: создать клиент-сервер.

**Исходный текст программы**

|  |
| --- |
| сlient.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("Can't 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, "help") == 0)  {  printf("Avalible commands:\n");  printf("exit - closes app\n");  printf("help - shows avalible commands\n");  printf("any other command will be interpreted as 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 lost\n");  exit(1);  }  else if (status == -1)  {  printf("File not found by server\n");  }  else  {  char filecon[1000];  printf("%s\n", status);  }  }  }  close(sockfd);  } |

|  |
| --- |
| 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)  {  long prev = ftell(fp);  fseek(fp, 0L, SEEK\_END);  long sz = ftell(fp);  fseek(fp, prev, 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) // создаем новый процесс  continue;  while (1) // обрабатываем запросы нового пользователя  {  char filename[MAX\_FILENAME\_SIZE];  int n = read(newsockfd, filename, MAX\_FILENAME\_SIZE);  {  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("Avalible commands:\n");  printf("exit - closes app\n");  printf("help - shows avalible commands\n");  }  else  {  printf("Unknown command, please use help to get list of avalible commands\n");  }  }  }  } |

**Вывод:**

В ходе выполнения лабораторной работы, я создал клиент-сервер.