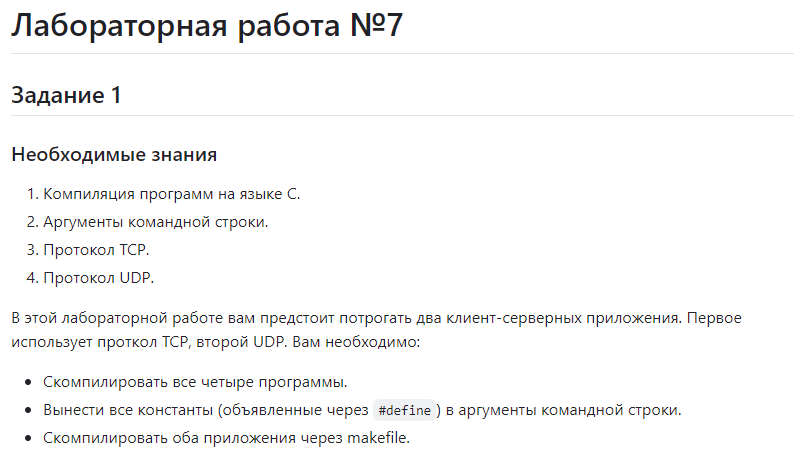
**Лабораторная работа 7**



**tcpclient.c:**

#include <arpa/inet.h>

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <unistd.h>

#define BUFSIZE 100

#define SADDR struct sockaddr

#define SIZE sizeof(struct sockaddr\_in)

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

  int fd;

  int nread;

  //char buf[BUFSIZE];

  struct sockaddr\_in servaddr;

  if (argc < 4) {

     printf("Usage: %s <IP address> <port> <buffer\_size>\n", argv[0]);

    exit(1);

  }

  int buffer\_size = atoi(argv[3]);

  char buf[buffer\_size];

  if ((fd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

    perror("socket creating");

    exit(1);

  }

  memset(&servaddr, 0, SIZE);

  servaddr.sin\_family = AF\_INET;

  if (inet\_pton(AF\_INET, argv[1], &servaddr.sin\_addr) <= 0) {

    perror("bad address");

    exit(1);

  }

  servaddr.sin\_port = htons(atoi(argv[2]));

  if (connect(fd, (SADDR \*)&servaddr, SIZE) < 0) {

    perror("connect");

    exit(1);

  }

  write(1, "Input message to send\n", 22);

  while ((nread = read(0, buf, buffer\_size)) > 0) {

    if (write(fd, buf, nread) < 0) {

      perror("write");

      exit(1);

    }

  }

  close(fd);

  exit(0);

}

**tcpserver.c:**

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <unistd.h>

#define SERV\_PORT 10050

#define BUFSIZE 100

#define SADDR struct sockaddr

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

  if (argc < 3) {

    printf("Usage: %s <port> <buffer\_size>\n", argv[0]);

    exit(1);

  }

  const size\_t kSize = sizeof(struct sockaddr\_in);

  int lfd, cfd;

  int nread;

  //char buf[BUFSIZE];

  int buffer\_size = atoi(argv[2]);

  char buf[buffer\_size];

  struct sockaddr\_in servaddr;

  struct sockaddr\_in cliaddr;

  if ((lfd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

    perror("socket");

    exit(1);

  }

  memset(&servaddr, 0, kSize);

  servaddr.sin\_family = AF\_INET;

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

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

  if (bind(lfd, (SADDR \*)&servaddr, kSize) < 0) {

    perror("bind");

    exit(1);

  }

  if (listen(lfd, 5) < 0) {

    perror("listen");

    exit(1);

  }

  while (1) {

    unsigned int clilen = kSize;

    if ((cfd = accept(lfd, (SADDR \*)&cliaddr, &clilen)) < 0) {

      perror("accept");

      exit(1);

    }

    printf("connection established\n");

    while ((nread = read(cfd, buf, buffer\_size)) > 0) {

      write(1, &buf, nread);

    }

    if (nread == -1) {

      perror("read");

      exit(1);

    }

    close(cfd);

  }

}

**udpclient.c:**  
  
#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <arpa/inet.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#define SERV\_PORT 20001

#define BUFSIZE 1024

#define SADDR struct sockaddr

#define SLEN sizeof(struct sockaddr\_in)

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

  int sockfd, n;

  //char sendline[BUFSIZE], recvline[BUFSIZE + 1];

  struct sockaddr\_in servaddr;

  //struct sockaddr\_in cliaddr;

  if (argc != 4) {

    printf("usage: %s <IPaddress of server> <port> <buffer\_size>\n", argv[0]);

    exit(1);

  }

  int buffer\_size = atoi(argv[3]);

  char sendline[buffer\_size], recvline[buffer\_size + 1];

  memset(&servaddr, 0, sizeof(servaddr));

  servaddr.sin\_family = AF\_INET;

  servaddr.sin\_port = htons(atoi(argv[2]));

  if (inet\_pton(AF\_INET, argv[1], &servaddr.sin\_addr) < 0) {

    perror("inet\_pton problem");

    exit(1);

  }

  if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0) {

    perror("socket problem");

    exit(1);

  }

  write(1, "Enter string\n", 13);

  while ((n = read(0, sendline, buffer\_size)) > 0) {

    if (sendto(sockfd, sendline, n, 0, (SADDR \*)&servaddr, SLEN) == -1) {

      perror("sendto problem");

      exit(1);

    }

    if (recvfrom(sockfd, recvline, buffer\_size, 0, NULL, NULL) == -1) {

      perror("recvfrom problem");

      exit(1);

    }

    printf("REPLY FROM SERVER= %s\n", recvline);

  }

  close(sockfd);

}

**udpserver.c:**

#include <arpa/inet.h>

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#define SERV\_PORT 20001

#define BUFSIZE 1024

#define SADDR struct sockaddr

#define SLEN sizeof(struct sockaddr\_in)

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

  if (argc < 3) {

    printf("Usage: %s <port> <buffer\_size>\n", argv[0]);

    exit(1);

  }

  int sockfd, n;

  //char mesg[BUFSIZE], ipadr[16];

  int buffer\_size = atoi(argv[2]);

  char mesg[buffer\_size], ipadr[16];

  struct sockaddr\_in servaddr;

  struct sockaddr\_in cliaddr;

  if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0) {

    perror("socket problem");

    exit(1);

  }

  memset(&servaddr, 0, SLEN);

  servaddr.sin\_family = AF\_INET;

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

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

  if (bind(sockfd, (SADDR \*)&servaddr, SLEN) < 0) {

    perror("bind problem");

    exit(1);

  }

  printf("SERVER starts...\n");

  while (1) {

    unsigned int len = SLEN;

    if ((n = recvfrom(sockfd, mesg, buffer\_size, 0, (SADDR \*)&cliaddr, &len)) < 0) {

      perror("recvfrom");

      exit(1);

    }

    mesg[n] = 0;

    printf("REQUEST %s      FROM %s : %d\n", mesg,

           inet\_ntop(AF\_INET, (void \*)&cliaddr.sin\_addr.s\_addr, ipadr, 16),

           ntohs(cliaddr.sin\_port));

    if (sendto(sockfd, mesg, n, 0, (SADDR \*)&cliaddr, len) < 0) {

      perror("sendto");

      exit(1);

    }

  }

}

**makefile:**

CC = gcc

CFLAGS = -Wall -g

all: tcpclient tcpserver udpclient udpserver

tcpclient: tcpclient.c

    $(CC) $(CFLAGS) -o tcpclient tcpclient.c

tcpserver: tcpserver.c

    $(CC) $(CFLAGS) -o tcpserver tcpserver.c

udpclient: udpclient.c

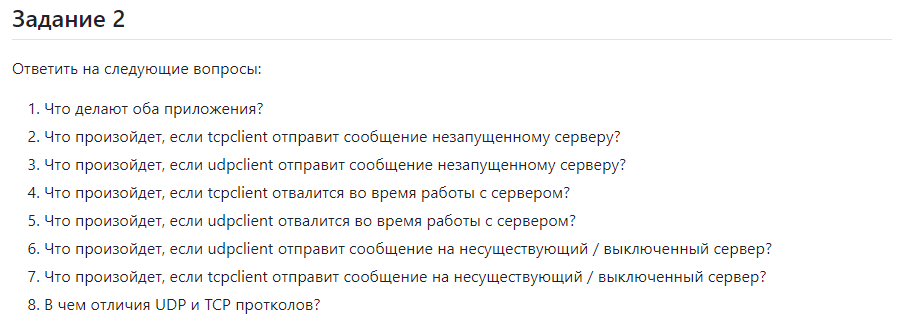
    $(CC) $(CFLAGS) -o udpclient udpclient.c

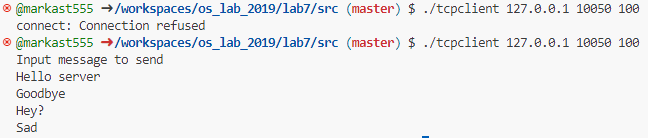
udpserver: udpserver.c

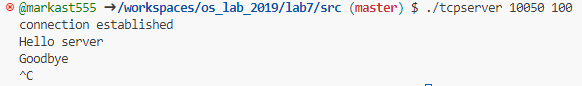
    $(CC) $(CFLAGS) -o udpserver udpserver.c

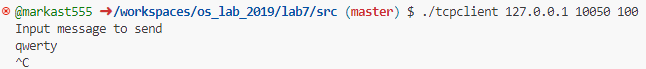
clean:

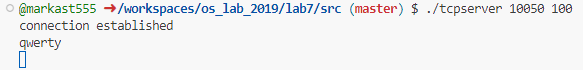
    rm -f tcpclient tcpserver udpclient udpserver

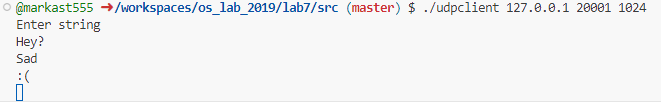


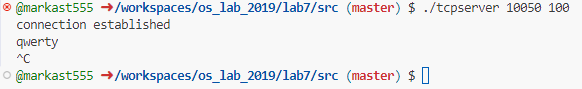


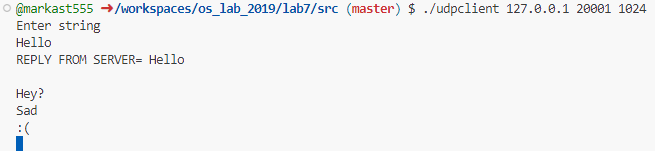


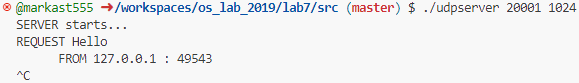


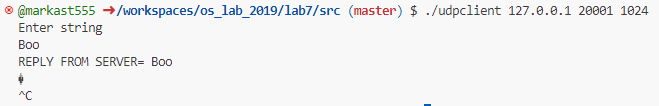


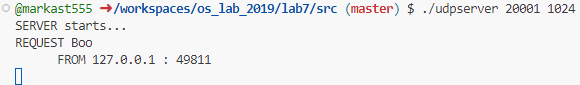














Эти приложения реализуют клиент-серверную архитектуру с использованием протоколов TCP и UDP.



Соединение с сервером не установится.



Сообщение не будет доставлено.



Ничего не произойдёт.



Ничего не произойдёт.



Сообщение не будет доставлено.



Соединение с сервером не установится.



**Соединение**:

**TCP**: Ориентированный на соединение. Устанавливается надежное соединение перед передачей данных.

**UDP**: Без соединения. Данные отправляются без предварительного установления соединения.

**Надежность**:

* + **TCP**: Гарантирует доставку данных, проверяет целостность и порядок. Если данные потеряны, они будут повторно отправлены.
  + **UDP**: Не гарантирует доставку, порядок или целостность данных. Нет механизма повторной отправки.

**Скорость**:

* + **TCP**: Более медленный из-за дополнительных проверок и установления соединения.
  + **UDP**: Более быстрый, так как не требует установления соединения и дополнительных проверок.

**Использование**:

* + **TCP**: Используется для приложений, требующих надежности, таких как веб-браузеры и электронная почта.
  + **UDP**: Используется для приложений, где скорость важнее надежности, таких как потоковое видео и онлайн-игры.