**Lab 9**

1. **TCP socket programming**

./tcpnnn 192.168.5.1 5555 10 alice.txt

./udpvvv 192.168.5.1 5005 10 alice.txt

TCP client

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CLIENT CODE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <stdio.h>  #include <sys/socket.h>  #include <netinet/in.h>  #include <string.h>  int main(int argc, char\* argv[]){  int clientSocket;  char buffer[1024];  struct sockaddr\_in serverAddr;  socklen\_t addr\_size;  /\*Open a socket and set it’s properties;\*/  clientSocket = socket(PF\_INET, SOCK\_STREAM, 0); // define ip, stream data, default protocol- TCP      /\*Input: IP\_address, port, num\_parts;\*/  serverAddr.sin\_family = AF\_INET; //Address family of the server = ipv4    serverAddr.sin\_port = htons(atoi(argv[2])); //Set port number, using htons to use proper byte order  serverAddr.sin\_addr.s\_addr = inet\_addr(argv[1]); //Set IP address to localhost    memset(serverAddr.sin\_zero, '\0', sizeof serverAddr.sin\_zero); //Set all bits in buff to 0  /\*Connect to server;\*/    addr\_size = sizeof serverAddr; // the struct contain ip and port num    if (connect(clientSocket, (struct sockaddr \*) &serverAddr, addr\_size) == -1){  printf("errorrrrr\n");  }  // connct to server: client sok, pointer to add server, add size      /\*Read the message from the server into the buffer\*/  int i=0;  while (i<atoi(argv[3])) {  int n = recv(clientSocket, buffer, 1024, 0); // n= num bytes read, write ti buffer, until 1024 byte each time  if (n == 0) break;  i++;  printf("%.\*s", n, buffer); //Print the received message  }    close(clientSocket); // close socket and free mem    return 0;  } |

Tcp server

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SERVER CODE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <stdio.h>  #include <sys/socket.h>  #include <netinet/in.h>  #include <string.h>  int main(int argc, char\* argv[]){  int welcomeSocket, newSocket;  char buffer[1024];  struct sockaddr\_in serverAddr;  struct sockaddr\_storage serverStorage;  socklen\_t addr\_size;  int num\_part = atoi(argv[3]);    /\*Open a socket and set it’s properties;\*/    welcomeSocket = socket(PF\_INET, SOCK\_STREAM, 0); // define ip, stream data, default protocol- TCP    /\*Input: IP\_address, port, num\_parts;\*/  serverAddr.sin\_family = AF\_INET; //Address family of the server = ipv4    serverAddr.sin\_port = htons(atoi(argv[2])); //Set port number, using htons to use proper byte order      serverAddr.sin\_addr.s\_addr = inet\_addr(argv[1]); //Set IP address to localhost    memset(serverAddr.sin\_zero, '\0', sizeof serverAddr.sin\_zero); //Set all bits in buff to 0  bind(welcomeSocket, (struct sockaddr \*) &serverAddr, sizeof(serverAddr)); //Bind the address struct to the socket    /\*Wait for a connection:\*/  if(listen(welcomeSocket,5)==0) // have a queue of 5 for listen  printf("Listening\n");  else  printf("Error\n");  /\* Accept call creates a new socket for the incoming connection\*/    addr\_size = sizeof serverStorage;  newSocket = accept(welcomeSocket, (struct sockaddr \*) &serverStorage, &addr\_size); // accept return a new FD to new soket.  /\*read from "alice" and send chunk of 1024 bytes over the socket\*/    FILE \*file;  unsigned long totalBytes = 0;    file = fopen(argv[4], "r");  if (file == NULL) {  printf("Could not open file");  return 1;  }    int i=0;  printf("%d", num\_part);  while(i<=num\_part){    totalBytes += fread(buffer, sizeof(char), 1024, file);  printf("chunk data (1024) read, total: %lu \n",totalBytes);  i++ ;  send(newSocket,buffer,1024,0); // send the buffer to client  }  printf("size byte sent: %lu\n", totalBytes);    fclose(file); // close file  close(newSocket); // close socket and free mem  return 0;} |

**UDP**

UDP client

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <sys/socket.h>  #include <netinet/in.h>  int main() {  int sockfd;  struct sockaddr\_in servaddr, cliaddr;  char buffer[1024];  int n;  socklen\_t len;  /\* Create socket \*/  sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);  /\* Bind socket \*/  memset(&servaddr, 0, sizeof(servaddr));  servaddr.sin\_family = AF\_INET;  servaddr.sin\_addr.s\_addr = inet\_addr("192.168.5.1"); //INADDR\_ANY  servaddr.sin\_port = htons(5005);  bind(sockfd, (struct sockaddr \*) &servaddr, sizeof(servaddr));  /\* Wait for data \*/  len = sizeof(cliaddr);  n = recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr \*) &cliaddr, &len);  buffer[n] = '\0';  printf("Received: %s\n", buffer);  /\* Close socket \*/  close(sockfd);  return 0;  } |

UDP server

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <sys/socket.h>  #include <netinet/in.h>  int main(int argc, char\* argv[]) {  int sockfd;  struct sockaddr\_in servaddr, cliaddr;  char buffer[1024];  int num\_part= atoi(argv[3]); // num part = 10  /\* Create socket \*/  sockfd = socket(AF\_INET, SOCK\_DGRAM, 0);  /\* Bind socket \*/  memset(&servaddr, 0, sizeof(servaddr));  servaddr.sin\_family = AF\_INET;  servaddr.sin\_addr.s\_addr = inet\_addr(argv[1]);  servaddr.sin\_port = htons(atoi(argv[2])); // port 5005  bind(sockfd, (struct sockaddr \*) &servaddr, sizeof(servaddr));  FILE \*file;  file = fopen(argv[4], "r");  unsigned long totalBytes = 0;  if (file == NULL) {  printf("Could not open file");  return 1;  }    int i=0;  while(i<=num\_part){    totalBytes += fread(buffer, sizeof(char), 1024, file);  printf("chunk data (1024) read, total: %lu \n",totalBytes);  i++ ;  /\* Send message \*/  sendto(sockfd, buffer, 1024, 0, (struct sockaddr \*) &servaddr, sizeof(servaddr));  }  printf("size byte sent: %lu\n", totalBytes);    fclose(file); // close file  close(sockfd); // close socket and free mem  return 0;  } |