Week 14 Lab 2:

Graphical user interface, text

Description automatically generated

echoServer1.c:

#include <stdlib.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <unistd.h>

#define MAXLINE 4096 \* 4 /\*max text line length\*/

#define SERV\_PORT 3000 /\*port\*/

#define LISTENQ 8 /\*maximum number of client connections \*/

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

**int** listenfd, connfd, n;

socklen\_t clilen;

**char** buf[MAXLINE];

**struct** sockaddr\_in cliaddr, servaddr;

pid\_t childpid;

//creation of the socket

listenfd = socket (AF\_INET, SOCK\_STREAM, 0);

//preparation of the socket address

servaddr.sin\_family = AF\_INET;

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

servaddr.sin\_port = htons(SERV\_PORT);

bind(listenfd, (**struct** sockaddr \*) &servaddr, **sizeof**(servaddr));

listen(listenfd, LISTENQ);

printf("%s\n","Server running...waiting for connections.");

**for** ( ; ; ) {

clilen = **sizeof**(cliaddr);

connfd = accept(listenfd, (**struct** sockaddr \*) &cliaddr, &clilen);

printf("%s\n","Received request...");

childpid = fork();

**if**(childpid == 0) {

printf("%s\n", "Child process created to handle request");

**while** ( (n = recv(connfd, buf, MAXLINE,0)) > 0) {

printf("\n%s","String received from and resent to the client: ");

puts(buf);

send(connfd, buf, n, 0);

}

close(listenfd);

}

**if** (n < 0) {

perror("Read error");

exit(1);

}

exit(0);

}

close(connfd);

}

echoClient1.c:

#include <stdlib.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <arpa/inet.h>

#define MAXLINE 4096 \* 4 /\*max text line length\*/

#define SERV\_PORT 3000 /\*port\*/

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

{

**int** sockfd;

**struct** sockaddr\_in servaddr;

**char** sendline[MAXLINE], recvline[MAXLINE];

//basic check of the arguments

//additional checks can be inserted

**if** (argc != 2) {

perror("Usage: TCPClient <IP address of the server");

exit(1);

}

//Create a socket for the client

//If sockfd<0 there was an error in the creation of the socket

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

perror("Problem in creating the socket");

exit(2);

}

//Creation of the socket

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

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr= inet\_addr(argv[1]);

servaddr.sin\_port = htons(SERV\_PORT); //convert to big-endian order

//Connection of the client to the socket

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

perror("Problem in connecting to the server");

exit(3);

}

**while** (fgets(sendline, MAXLINE, stdin) != **NULL**) {

send(sockfd, sendline, strlen(sendline), 0);

**if** (recv(sockfd, recvline, MAXLINE,0) == 0){

//error: server terminated prematurely

perror("The server terminated prematurely");

exit(4);

}

printf("%s", "String received from the server: ");

fputs(recvline, stdout);

printf("\n");

}

exit(0);

}