```
/*--------server.c-------
sample code*/

// Standard libraries
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Socket-related libraries
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

#define MAX_LINE_SIZE 50
#define LISTEN_QUEUE 5

void print_usage(){
  fprintf(stderr, "Usage: ./server [port]\n");
  fprintf(stderr, "Example: ./server 4845\n\n");
}

void check_args(int argc, char *argv[]){
  if( argc != 2 ) {
    print_usage();
        exit(1);
  }
}

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

  struct addrinfo hints;
  struct addrinfo *serverInfo;

  // Check arguments and exit if they are not well constructed
  check_args(argc, argv);

  // Fill the 'hints' structure
  memset( &hints, 0, sizeof(hints) );
  hints.ai_flags    = AI_PASSIVE;  // Use the local IP address
  hints.ai_family   = AF_INET;     // Use IPv4 address
  hints.ai_socktype = SOCK_STREAM; // TCP sockets

  // Get the address information of the server ('NULL' uses the local address
)
  getaddrinfo( NULL , argv[1], &hints, &serverInfo );

  // Make a socket (identified with a socket file descriptor)
  int socketFD = socket( serverInfo->ai_family, serverInfo->ai_socktype,
serverInfo->ai_protocol );

  // Bind to the specified port (in getaddrinfo())
  bind( socketFD, serverInfo->ai_addr, serverInfo->ai_addrlen );

  // This is only used to be able to reuse the same port
```

```
  int yes = 1;
  setsockopt( socketFD, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof(int) );

  // Start to listen (only for TCP)
  listen(socketFD, LISTEN_QUEUE);
  printf("Listening in port %s...", argv[1]);
  fflush(stdout);

  // Free the address info struct
  freeaddrinfo(serverInfo);

  // Accept a connections (only for TCP) and create a socket for each one
  struct sockaddr_storage clientAddr;
  socklen_t clientAddr_size = sizeof(clientAddr);
  int recvFD;
  while (1) {
    recvFD = accept(socketFD, (struct sockaddr *)&clientAddr,
&clientAddr_size);

    // Receive data
    char line[MAX_LINE_SIZE];
    int num_bytes_received = recv(recvFD, line, MAX_LINE_SIZE-1, 0);

    // Print received data
    printf("\n   Received String: '%s'\n\n", line);
    printf("Listening in port %s...", argv[1]);
    fflush(stdout);

    // Close the socket associated to this client
    close(recvFD);
  }
  // Close the socket that was used to listen
  close(socketFD);

  return EXIT_SUCCESS;
}
```