Cliente / Servidor Sockets

Gustavo Bacagine
Kaique Ryan

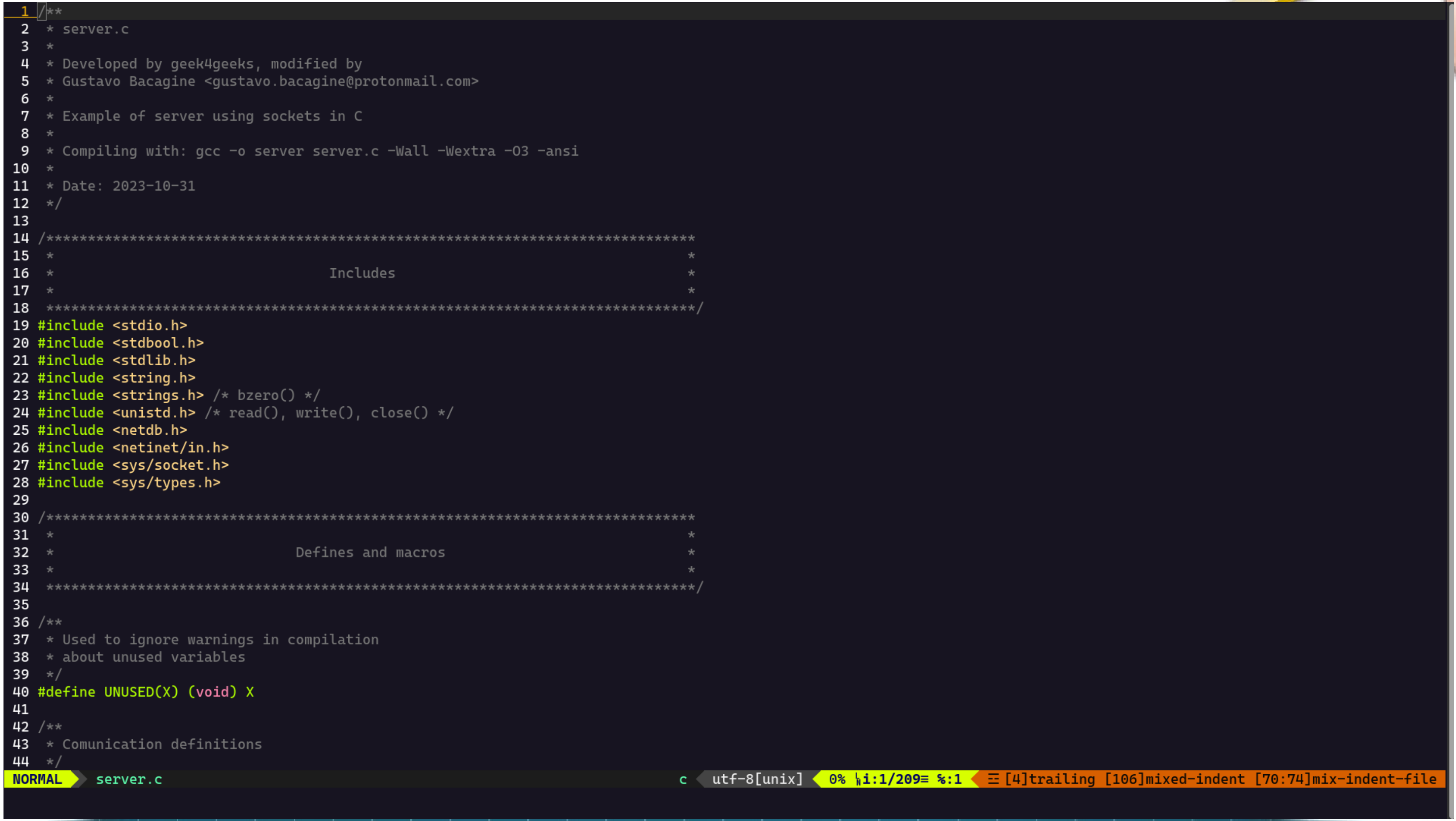
Modelo Cliente/Servidor

- Separa o programa em cliente e servidor como programas distintos para que se comunicam dentro da mesma rede

Bibliotecas e funções de socket em C

- sys/socket.h: Biblioteca C para manipulação de socket
- socket(int domain, int type, int protocol): Cria um novo socket
- bind(int socket, const struct sockaddr *address, socklen_t address_len): Associa o socket a um endereço
- listen(int socket, int backlog): Coloca o socket no modo de escuta
- accepet(int socket, struct sockaddr *restrict address, socklen_t *restrict address_len): Aceita uma nova conexão
- connect(int socket, const struct sockaddr *address, socklen_t address_len)): Conecta o socket a um endereço

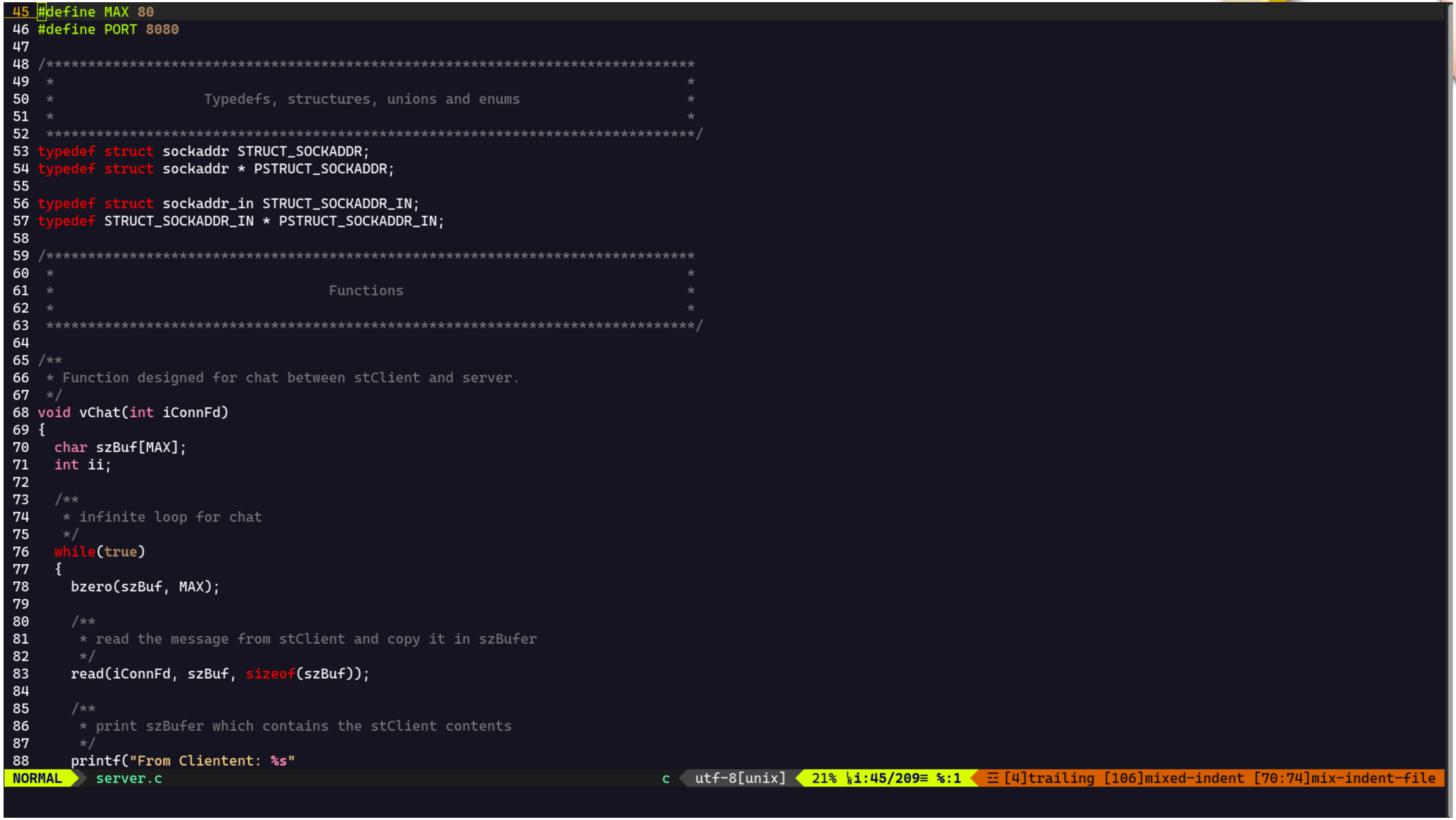
Exemplo em C - Servidor



```c
/**
 * server.c
 *
 * Developed by geek4geeks, modified by
 * Gustavo Bacagine <gustavo.bacagine@protonmail.com>
 *
 * Example of server using sockets in C
 *
 * Compiling with: gcc -o server server.c -Wall -Wextra -O3 -ansi
 *
 * Date: 2023-10-31
 */

/******************************************************************************
 *                                                                            *
 *                                 Includes                                   *
 *                                                                            *
 ******************************************************************************/
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h> /* bzero() */
#include <unistd.h> /* read(), write(), close() */
#include <netdb.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/types.h>

/******************************************************************************
 *                                                                            *
 *                             Defines and macros                             *
 *                                                                            *
 ******************************************************************************/

/**
 * Used to ignore warnings in compilation
 * about unused variables
 */
#define UNUSED(X) (void) X

/**
 * Comunication definitions
 */
```

```
NORMAL  server.c                    c  utf-8[unix]  0% ␤i:1/209≡ %:1  ☲ [4]trailing [106]mixed-indent [70:74]mix-indent-file
```

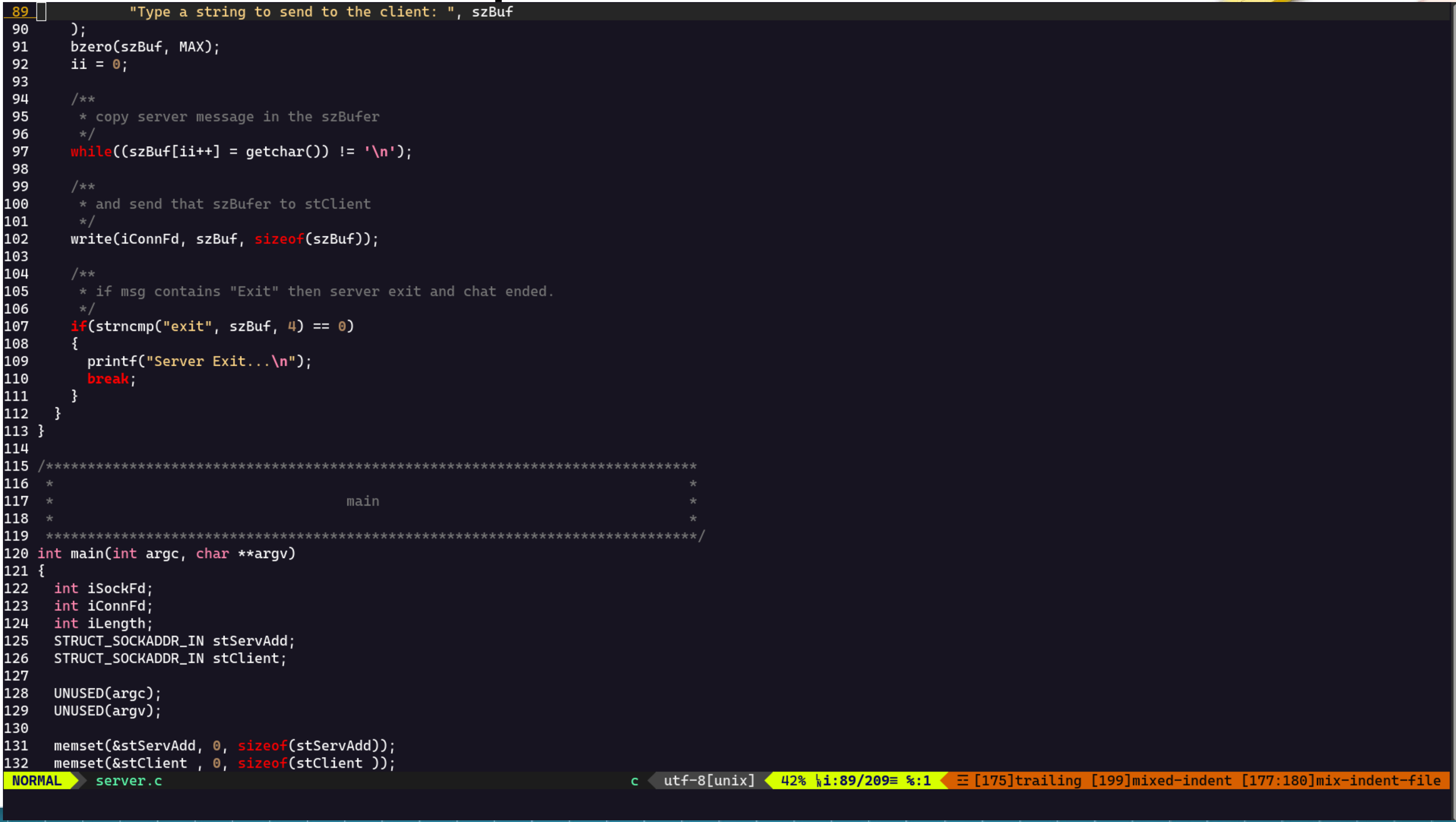
Exemplo em C - Servidor



```c
#define MAX 80
#define PORT 8080

/*******************************************************************************
 *                                                                            *
 *                  Typedefs, structures, unions and enums                    *
 *                                                                            *
 ******************************************************************************/
typedef struct sockaddr STRUCT_SOCKADDR;
typedef struct sockaddr * PSTRUCT_SOCKADDR;

typedef struct sockaddr_in STRUCT_SOCKADDR_IN;
typedef STRUCT_SOCKADDR_IN * PSTRUCT_SOCKADDR_IN;

/*******************************************************************************
 *                                                                            *
 *                                 Functions                                  *
 *                                                                            *
 ******************************************************************************/

/**
 * Function designed for chat between stClient and server.
 */
void vChat(int iConnFd)
{
  char szBuf[MAX];
  int ii;

  /**
   * infinite loop for chat
   */
  while(true)
  {
    bzero(szBuf, MAX);

    /**
     * read the message from stClient and copy it in szBufer
     */
    read(iConnFd, szBuf, sizeof(szBuf));

    /**
     * print szBufer which contains the stClient contents
     */
    printf("From Clientent: %s"
```

```
NORMAL  server.c                                  c  utf-8[unix]  21% ㏑:45/209≡ %:1  ☲ [4]trailing [106]mixed-indent [70:74]mix-indent-file
```

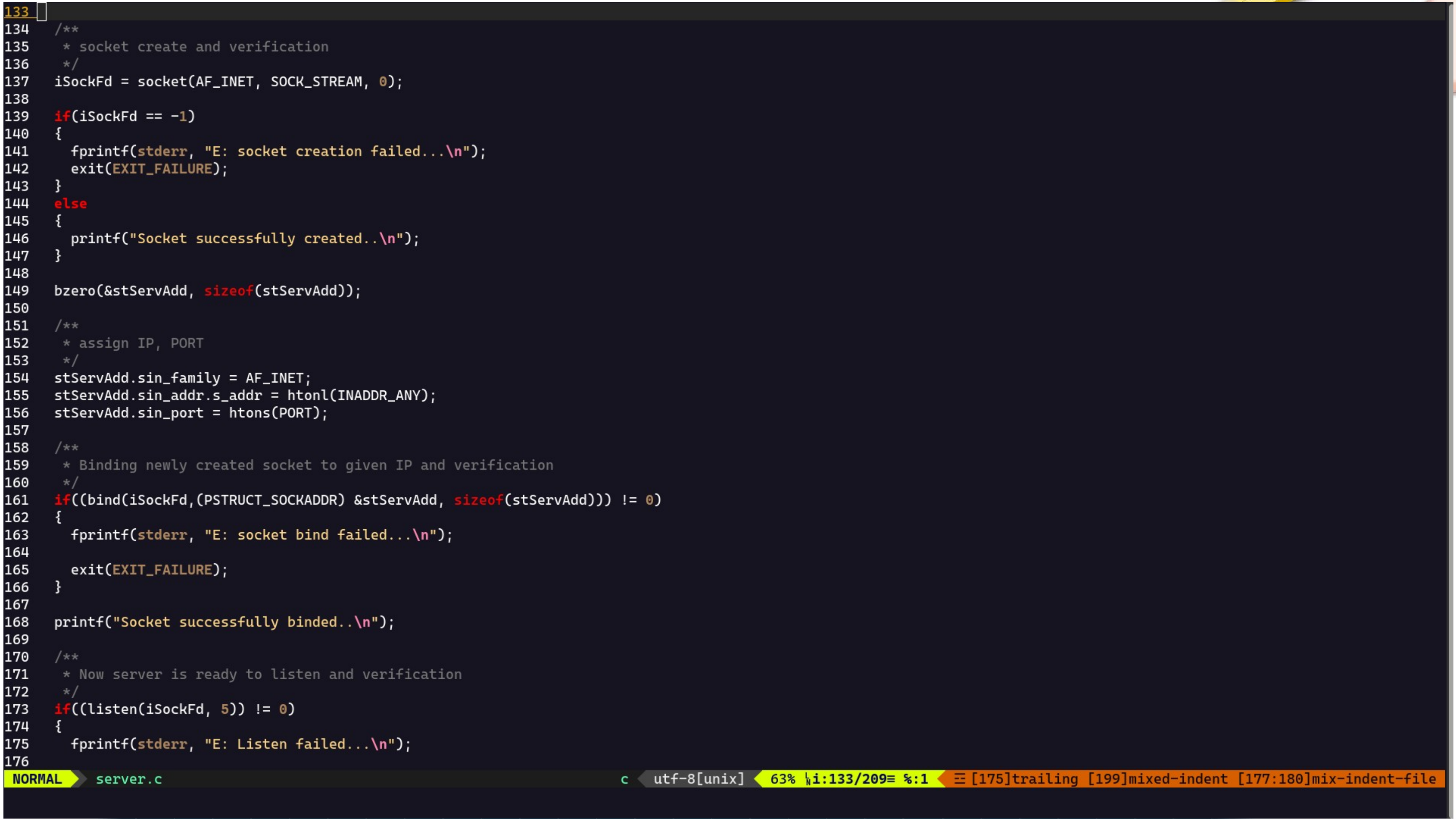
Exemplo em C - Servidor



```c
           "Type a string to send to the client: ", szBuf
    );
    bzero(szBuf, MAX);
    ii = 0;

    /**
     * copy server message in the szBufer
     */
    while((szBuf[ii++] = getchar()) != '\n');

    /**
     * and send that szBufer to stClient
     */
    write(iConnFd, szBuf, sizeof(szBuf));

    /**
     * if msg contains "Exit" then server exit and chat ended.
     */
    if(strncmp("exit", szBuf, 4) == 0)
    {
      printf("Server Exit...\n");
      break;
    }
  }
}

/******************************************************************************
 *                                    main                                    *
 *                                                                            *
 *                                                                            *
 ******************************************************************************/
int main(int argc, char **argv)
{
  int iSockFd;
  int iConnFd;
  int iLength;
  STRUCT_SOCKADDR_IN stServAdd;
  STRUCT_SOCKADDR_IN stClient;

  UNUSED(argc);
  UNUSED(argv);

  memset(&stServAdd, 0, sizeof(stServAdd));
  memset(&stClient , 0, sizeof(stClient ));
```

```
NORMAL  server.c                                  c  utf-8[unix]  42% ㏑:89/209☰ %:1  ☲ [175]trailing [199]mixed-indent [177:180]mix-indent-file
```

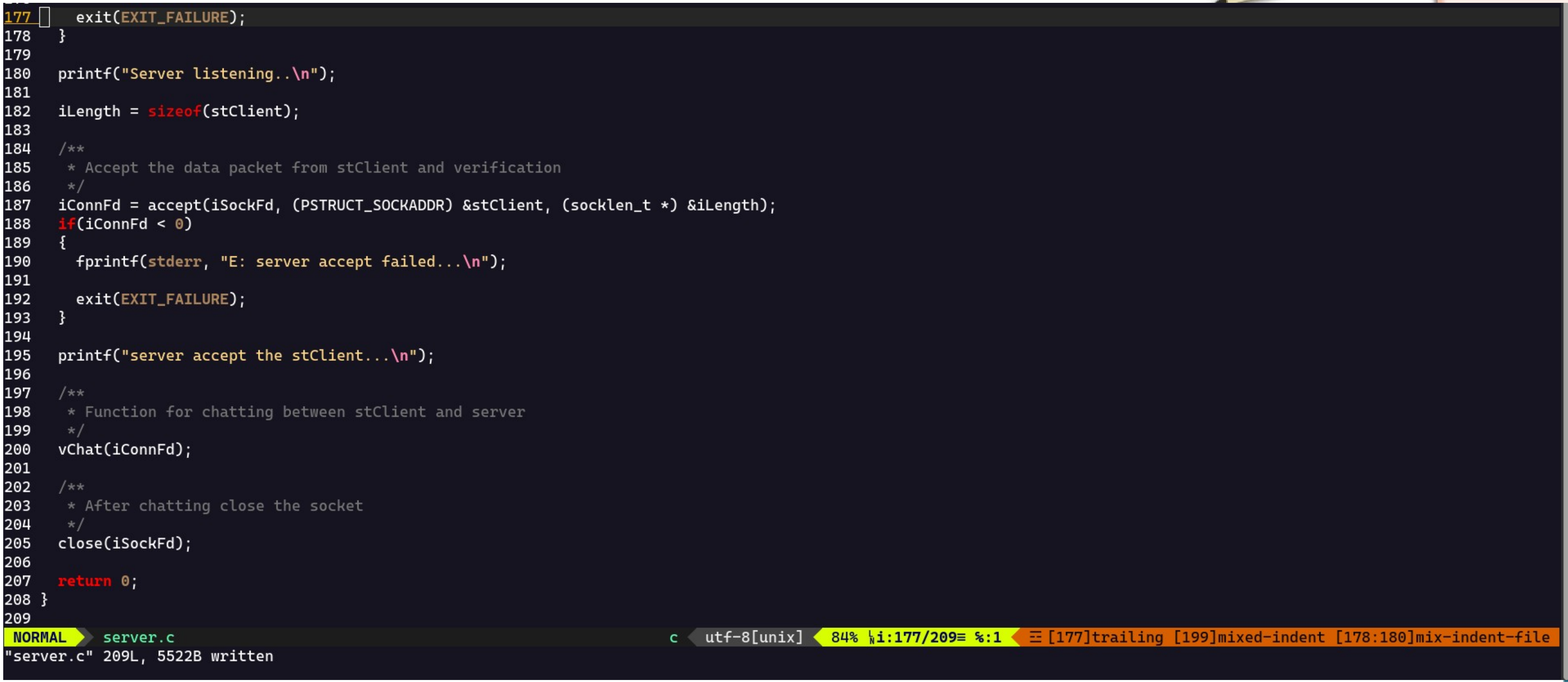
Exemplo em C - Servidor



```c

/**
 * socket create and verification
 */
iSockFd = socket(AF_INET, SOCK_STREAM, 0);

if(iSockFd == -1)
{
  fprintf(stderr, "E: socket creation failed...\n");
  exit(EXIT_FAILURE);
}
else
{
  printf("Socket successfully created..\n");
}

bzero(&stServAdd, sizeof(stServAdd));

/**
 * assign IP, PORT
 */
stServAdd.sin_family = AF_INET;
stServAdd.sin_addr.s_addr = htonl(INADDR_ANY);
stServAdd.sin_port = htons(PORT);

/**
 * Binding newly created socket to given IP and verification
 */
if((bind(iSockFd,(PSTRUCT_SOCKADDR) &stServAdd, sizeof(stServAdd))) != 0)
{
  fprintf(stderr, "E: socket bind failed...\n");

  exit(EXIT_FAILURE);
}

printf("Socket successfully binded..\n");

/**
 * Now server is ready to listen and verification
 */
if((listen(iSockFd, 5)) != 0)
{
  fprintf(stderr, "E: Listen failed...\n");

```

```
NORMAL  server.c                    c  utf-8[unix]  63%  ℕi:133/209≡ %:1  ☰ [175]trailing [199]mixed-indent [177:180]mix-indent-file
```

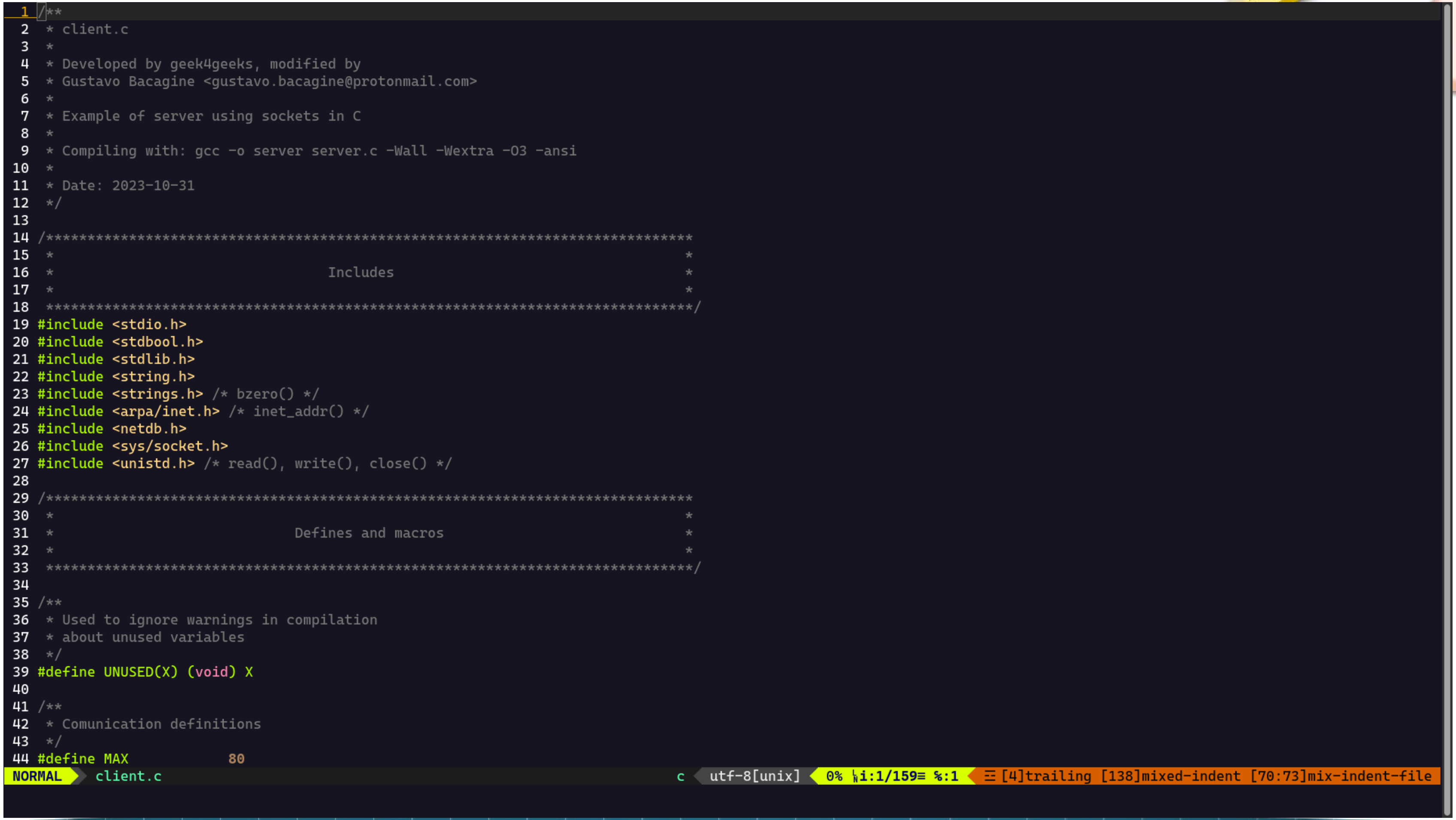
Exemplo em C - Servidor



```c
    exit(EXIT_FAILURE);
  }

  printf("Server listening..\n");

  iLength = sizeof(stClient);

  /**
   * Accept the data packet from stClient and verification
   */
  iConnFd = accept(iSockFd, (PSTRUCT_SOCKADDR) &stClient, (socklen_t *) &iLength);
  if(iConnFd < 0)
  {
    fprintf(stderr, "E: server accept failed...\n");

    exit(EXIT_FAILURE);
  }

  printf("server accept the stClient...\n");

  /**
   * Function for chatting between stClient and server
   */
  vChat(iConnFd);

  /**
   * After chatting close the socket
   */
  close(iSockFd);

  return 0;
}
```

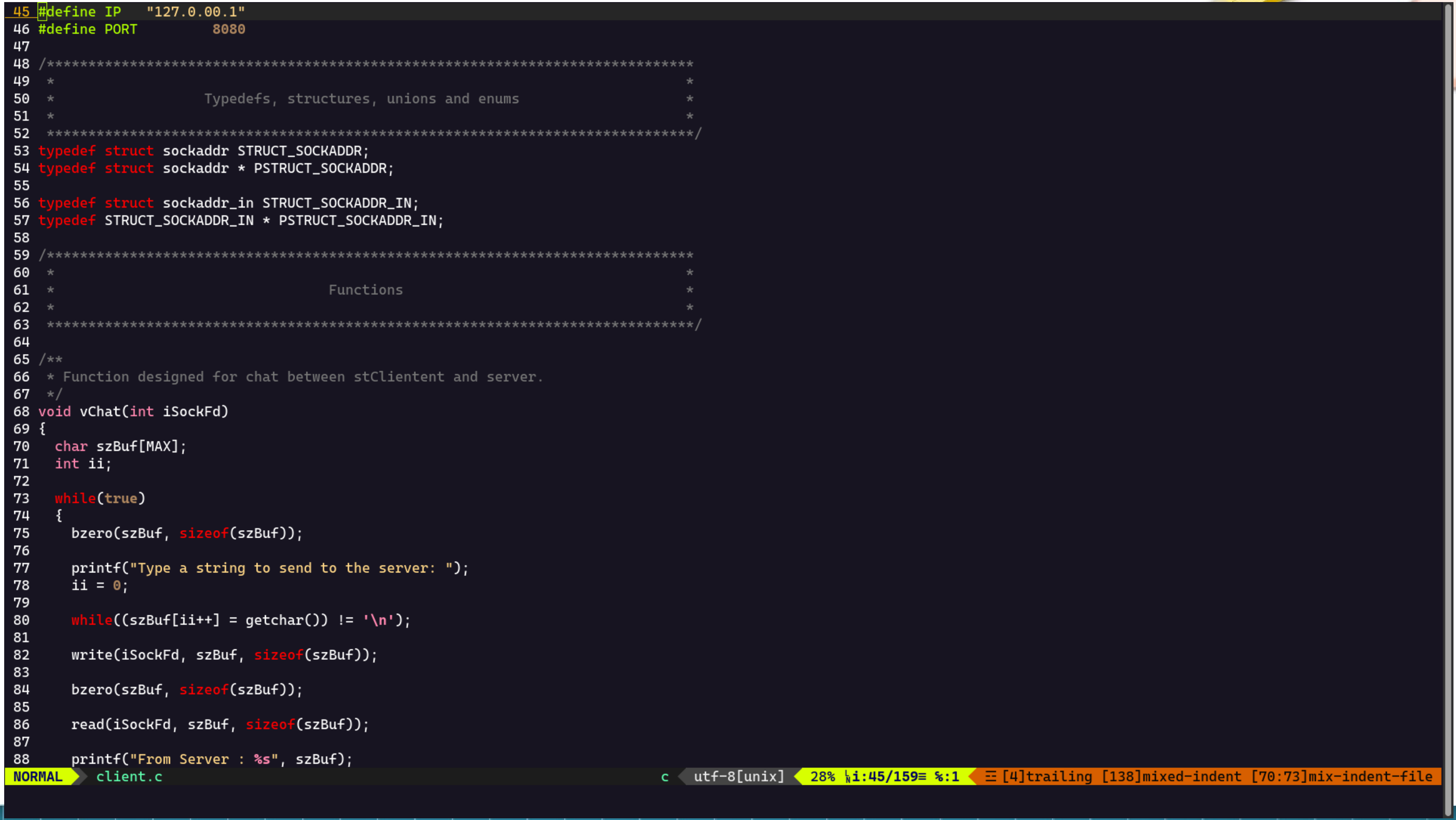
```
NORMAL  server.c                                   c  utf-8[unix]  84% ℏi:177/209≡ %:1  ☲ [177]trailing [199]mixed-indent [178:180]mix-indent-file
"server.c" 209L, 5522B written
```

Exemplo em C - Cliente



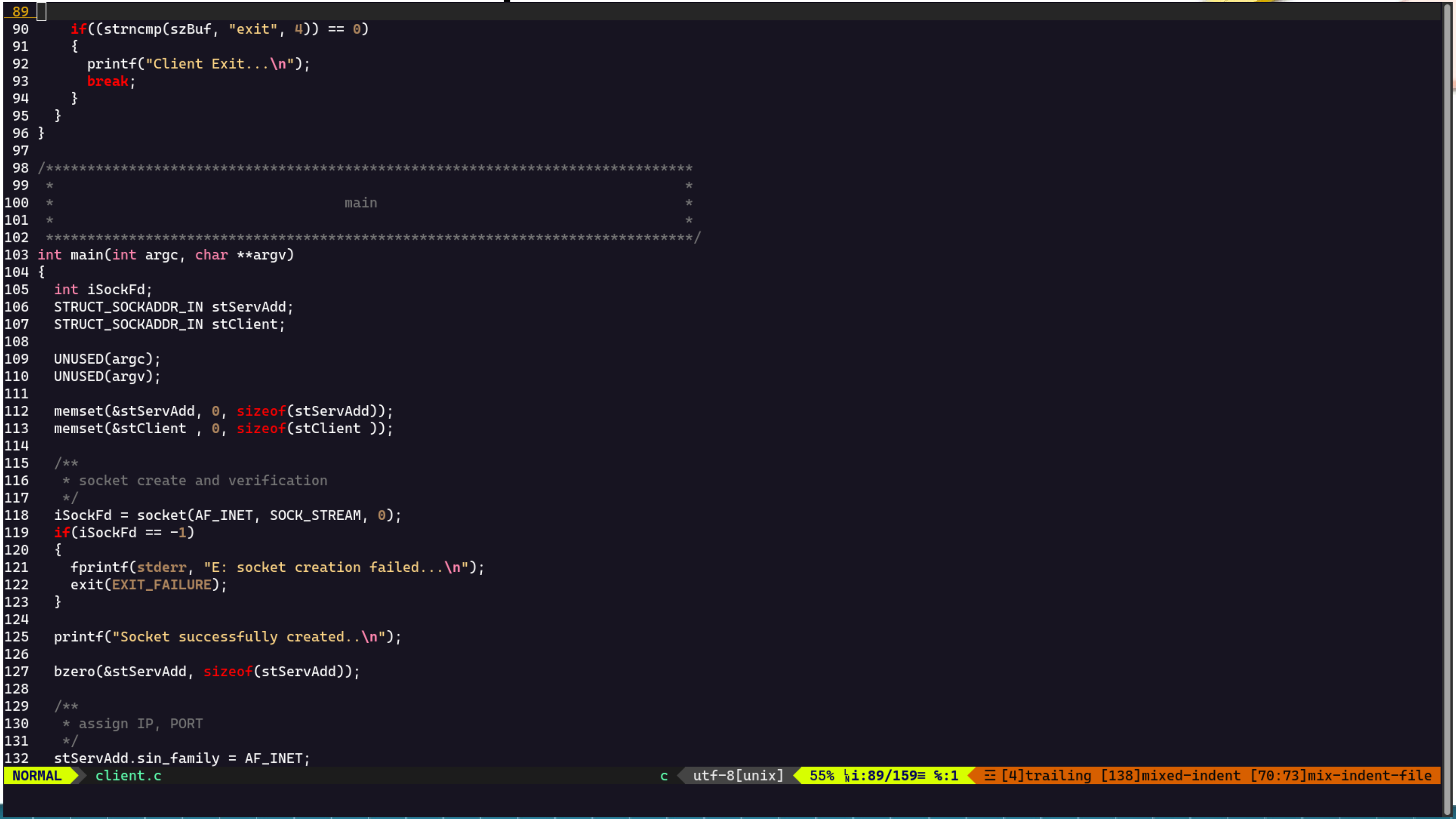
```
/**
 * client.c
 *
 * Developed by geek4geeks, modified by
 * Gustavo Bacagine <gustavo.bacagine@protonmail.com>
 *
 * Example of server using sockets in C
 *
 * Compiling with: gcc -o server server.c -Wall -Wextra -O3 -ansi
 *
 * Date: 2023-10-31
 */

/******************************************************************************
 *                                                                            *
 *                                 Includes                                   *
 *                                                                            *
 ******************************************************************************/
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>
#include <strings.h> /* bzero() */
#include <arpa/inet.h> /* inet_addr() */
#include <netdb.h>
#include <sys/socket.h>
#include <unistd.h> /* read(), write(), close() */

/******************************************************************************
 *                                                                            *
 *                             Defines and macros                             *
 *                                                                            *
 ******************************************************************************/

/**
 * Used to ignore warnings in compilation
 * about unused variables
 */
#define UNUSED(X) (void) X

/**
 * Comunication definitions
 */
#define MAX            80
NORMAL  client.c                                                        c  utf-8[unix]  0% ㏑:1/159≡ %:1  ☲ [4]trailing [138]mixed-indent [70:73]mix-indent-file
```

Exemplo em C - Cliente



```
#define IP   "127.0.00.1"
#define PORT          8080

/******************************************************************************
 *                                                                            *
 *                   Typedefs, structures, unions and enums                   *
 *                                                                            *
 ******************************************************************************/
typedef struct sockaddr STRUCT_SOCKADDR;
typedef struct sockaddr * PSTRUCT_SOCKADDR;

typedef struct sockaddr_in STRUCT_SOCKADDR_IN;
typedef STRUCT_SOCKADDR_IN * PSTRUCT_SOCKADDR_IN;

/******************************************************************************
 *                                                                            *
 *                                  Functions                                 *
 *                                                                            *
 ******************************************************************************/

/**
 * Function designed for chat between stClientent and server.
 */
void vChat(int iSockFd)
{
  char szBuf[MAX];
  int ii;

  while(true)
  {
    bzero(szBuf, sizeof(szBuf));

    printf("Type a string to send to the server: ");
    ii = 0;

    while((szBuf[ii++] = getchar()) != '\n');

    write(iSockFd, szBuf, sizeof(szBuf));

    bzero(szBuf, sizeof(szBuf));

    read(iSockFd, szBuf, sizeof(szBuf));

    printf("From Server : %s", szBuf);
NORMAL  client.c                                          c  utf-8[unix]  28% ㏑:45/159≡ ℅:1  ☲ [4]trailing [138]mixed-indent [70:73]mix-indent-file
```

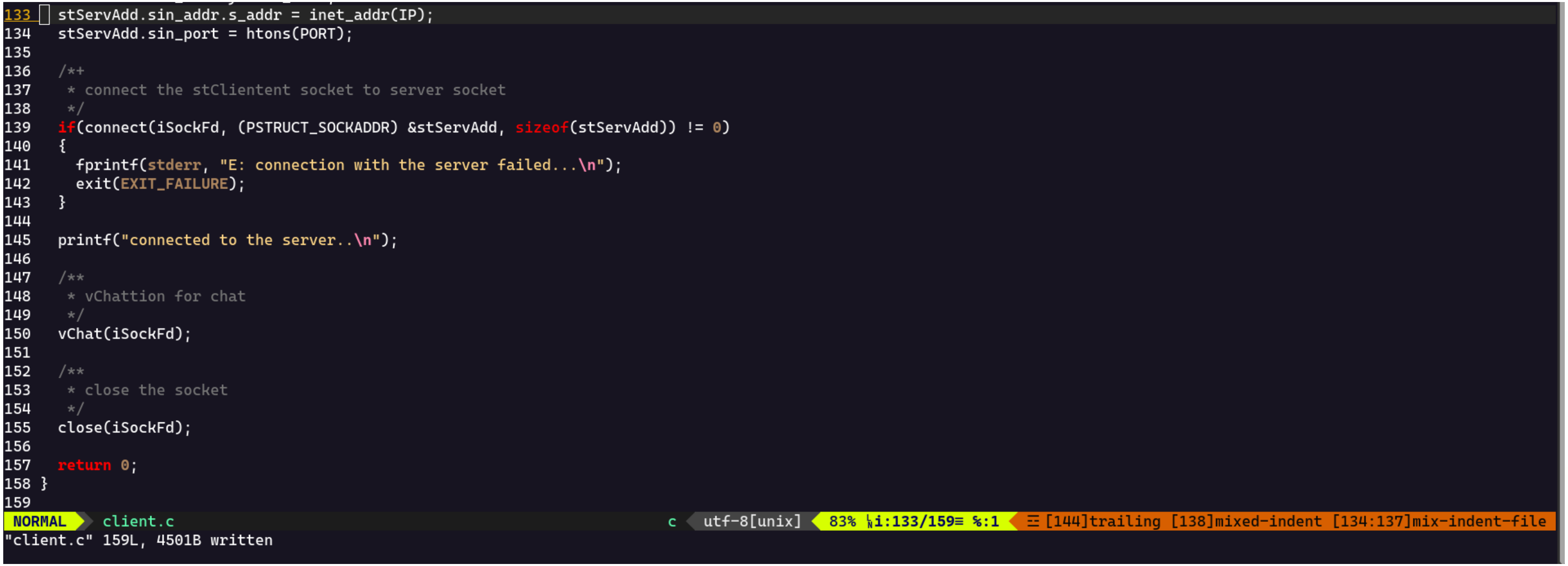
Exemplo em C - Cliente



```c

    if((strncmp(szBuf, "exit", 4)) == 0)
    {
      printf("Client Exit...\n");
      break;
    }
  }
}

/*******************************************************************************
 *                                                                             *
 *                                   main                                      *
 *                                                                             *
 *******************************************************************************/
int main(int argc, char **argv)
{
  int iSockFd;
  STRUCT_SOCKADDR_IN stServAdd;
  STRUCT_SOCKADDR_IN stClient;

  UNUSED(argc);
  UNUSED(argv);

  memset(&stServAdd, 0, sizeof(stServAdd));
  memset(&stClient , 0, sizeof(stClient ));

  /**
   * socket create and verification
   */
  iSockFd = socket(AF_INET, SOCK_STREAM, 0);
  if(iSockFd == -1)
  {
    fprintf(stderr, "E: socket creation failed...\n");
    exit(EXIT_FAILURE);
  }

  printf("Socket successfully created..\n");

  bzero(&stServAdd, sizeof(stServAdd));

  /**
   * assign IP, PORT
   */
  stServAdd.sin_family = AF_INET;
```

```
NORMAL  client.c                                          c  utf-8[unix]  55% ㏑:89/159≡ ℅:1  ☲ [4]trailing [138]mixed-indent [70:73]mix-indent-file
```

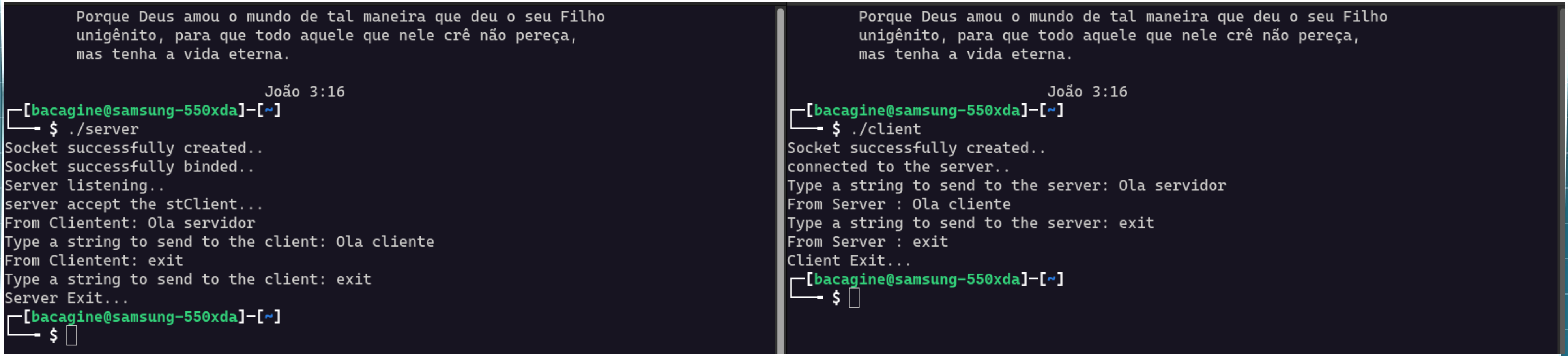
Exemplo em C - Cliente



```c
  stServAdd.sin_addr.s_addr = inet_addr(IP);
  stServAdd.sin_port = htons(PORT);

  /**
   * connect the stClientent socket to server socket
   */
  if(connect(iSockFd, (PSTRUCT_SOCKADDR) &stServAdd, sizeof(stServAdd)) != 0)
  {
    fprintf(stderr, "E: connection with the server failed...\n");
    exit(EXIT_FAILURE);
  }

  printf("connected to the server..\n");

  /**
   * vChattion for chat
   */
  vChat(iSockFd);

  /**
   * close the socket
   */
  close(iSockFd);

  return 0;
}

```

```
"client.c" 159L, 4501B written
```

Saída do exemplo em C



```
        Porque Deus amou o mundo de tal maneira que deu o seu Filho
        unigênito, para que todo aquele que nele crê não pereça,
        mas tenha a vida eterna.

                             João 3:16
┌─[bacagine@samsung-550xda]─[~]
└──╼ $ ./server
Socket successfully created..
Socket successfully binded..
Server listening..
server accept the stClient...
From Clientent: Ola servidor
Type a string to send to the client: Ola cliente
From Clientent: exit
Type a string to send to the client: exit
Server Exit...
┌─[bacagine@samsung-550xda]─[~]
└──╼ $
```

```
        Porque Deus amou o mundo de tal maneira que deu o seu Filho
        unigênito, para que todo aquele que nele crê não pereça,
        mas tenha a vida eterna.

                             João 3:16
┌─[bacagine@samsung-550xda]─[~]
└──╼ $ ./client
Socket successfully created..
connected to the server..
Type a string to send to the server: Ola servidor
From Server : Ola cliente
Type a string to send to the server: exit
From Server : exit
Client Exit...
┌─[bacagine@samsung-550xda]─[~]
└──╼ $
```

Fontes

- https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/
- Manpages do linux sobre as funções em C
- Slide “Modelo Cliente/Servidor em Sistemas Distribuídos” do professor Walter Matheos