

## Exercício 2

**Aluno:** Caio Augusto Alves Nolasco

**RA:** 195181

Instituto de Computação

Universidade Estadual de Campinas

Campinas, 27 de Outubro de 2020.

# Sumário

1	Questão 1 . . . . .	2
2	Questão 2 . . . . .	2
3	Questão 3 . . . . .	5
4	Questão 4 . . . . .	5
5	Questão 6 . . . . .	5
6	Questão 7 . . . . .	6
7	Questão 8 . . . . .	7

## 1 Questão 1

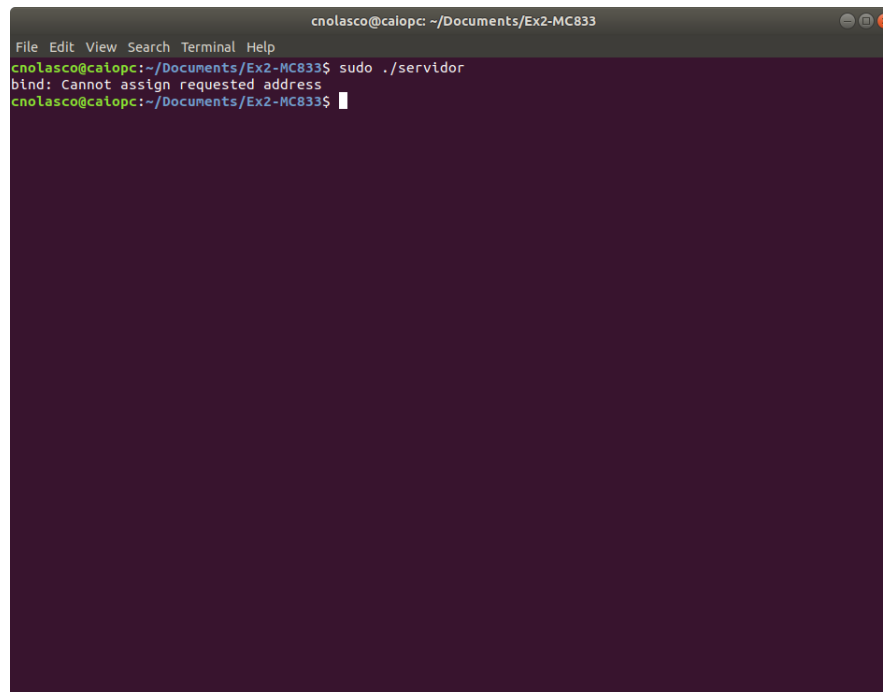
Para o código do cliente, os principais métodos usados para a comunicação via socket são: um struct *sockaddrin*, que armazena os endereços de internet e de família do cliente, assim como o número da porta; A chamada *socket* é responsável por criar o socket propriamente, e recebe como parâmetros o endereço de domínio AF\_INET, que especifica o domínio da Internet, o tipo de socket SOCK\_STREAM, informando que o socket é de fluxo de bytes e não de datagramas, e o terceiro argumento identifica o protocolo seguido para conexão, nesse caso, TCP; Os valores do struct *sockaddrin* são preenchidos com os valores pertinentes do cliente; Por fim, a função *connect* estabelece a conexão com o servidor, recebendo como parâmetros o socket criado por *socket()*, o endereço do servidor, com o número da porta, e o tamanho desse endereço.

Já no lado do servidor, o socket é criado da mesma maneira do que no cliente; após criar o socket, chama-se a função *bind*, que une o socket ao endereço do servidor. Como parâmetros são enviados o socket, o endereço em questão, e o tamanho deste endereço; A função *listen* faz com que o processo escute o socket por conexões. Seus parâmetros são o socket e o número de conexões suportadas pelo processo. Enfim, a função *accept* aceita a primeira conexão na fila de conexões, criando um novo socket que cuidará da conexão estabelecida com o cliente. Seus argumentos são o socket original, o endereço do cliente e o tamanho desse endereço.

## 2 Questão 2

Quando cliente.c e servidor.c são compilados e rodados na mesma máquina, o servidor não consegue definir o endereço de IP dado como um

valor arbitrário.

A terminal window titled 'cnolasco@calopc: ~/Documents/Ex2-MC833' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is 'cnolasco@calopc:~/Documents/Ex2-MC833\$'. The command 'sudo ./servidor' has been executed, resulting in the error message 'bind: Cannot assign requested address'. The prompt is now 'cnolasco@calopc:~/Documents/Ex2-MC833\$' with a cursor.

```
cnolasco@calopc: ~/Documents/Ex2-MC833
File Edit View Search Terminal Help
cnolasco@calopc:~/Documents/Ex2-MC833$ sudo ./servidor
bind: Cannot assign requested address
cnolasco@calopc:~/Documents/Ex2-MC833$
```

O erro pode ser corrigido colocando o endereço de IP do servidor como o endereço da máquina que roda os programas. Assim, a linha *servaddr.sinaddr.saddr = htonl("192.168.0.16")*; fica *servaddr.sinaddr.saddr = INADDRANY*;

```
cnolasco@calopc: ~/Documents/Ex2-MC833
File Edit View Search Terminal Help
cnolasco@calopc:~/Documents/Ex2-MC833$ sudo ./cliente 127.0.0.1
Mon Oct 26 04:19:37 2020
cnolasco@calopc:~/Documents/Ex2-MC833$
```

```
cnolasco@calopc: ~/Documents/Ex2-MC833
File Edit View Search Terminal Help
cnolasco@calopc:~/Documents/Ex2-MC833$ sudo ./servidor

```

### 3 Questão 3

Para realizar as mudanças pedidas no enunciado, muda-se os valores dos valores do struct *sockaddr\_in* em *servidor.c* para os valores mostrados na imagem seguinte.

```
servaddr.sin_family    = AF_INET;
servaddr.sin_addr.s_addr = INADDR_ANY;
servaddr.sin_port      = 0;
```

### 4 Questão 4

Conexões futuras ao servido em execução não precisam realizar chamadas novamente para criação de socket, já que ele já existe. Então, apenas a syscall *accept()* é invocada para uma nova conexão, para aceitar o pedido de uma nova conexão.

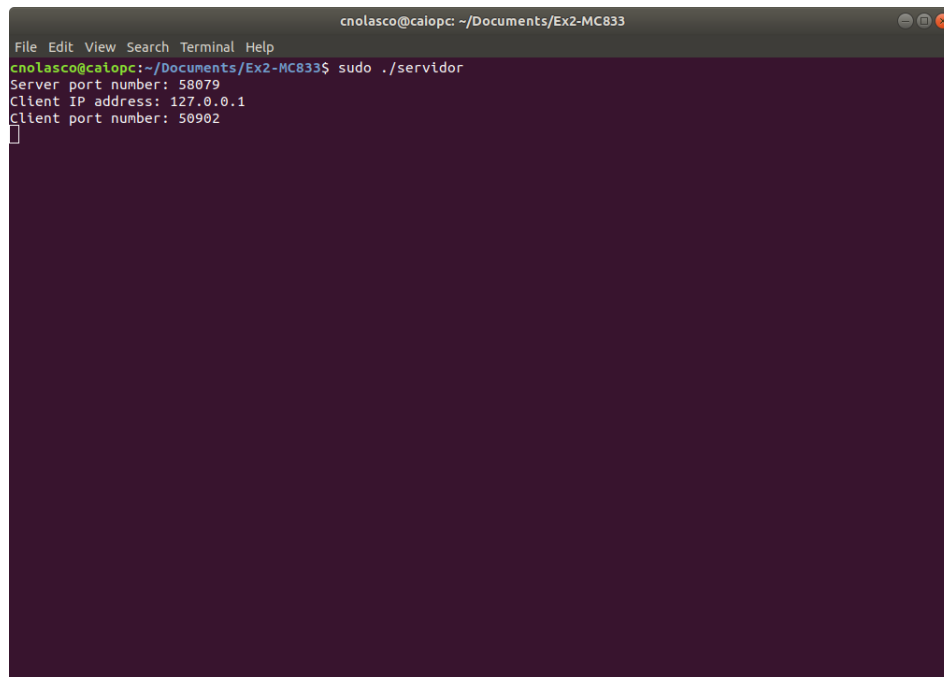
### 5 Questão 6

```
socklen_t len = sizeof(servaddr);
if (getsockname(sockfd, (struct sockaddr *)&servaddr, &len) == -1)
    perror("getsockname");
else
{
    char clientIP[16];
    inet_ntop(AF_INET, &servaddr.sin_addr, clientIP, sizeof(clientIP));
    printf("Client IP address: %s\n", clientIP);
    printf("Client port number: %d\n", ntohs(servaddr.sin_port));
}
```

```
cnoiasco@calopc: ~/Documents/Ex2-MC833
File Edit View Search Terminal Help
cnoiasco@calopc:~/Documents/Ex2-MC833$ sudo ./cliente 127.0.0.1
Client IP address: 127.0.0.1
Client port number: 33106
Mon Oct 26 16:19:47 2020
cnoiasco@calopc:~/Documents/Ex2-MC833$
```

## 6 Questão 7

```
socklen_t len = sizeof(servaddr);
if (getpeername(connfd, (struct sockaddr *)&servaddr, &len) == -1)
    perror("getpeername");
else
{
    char clientIP[16];
    inet_ntop(AF_INET, &servaddr.sin_addr, clientIP, sizeof(clientIP));
    printf("Client IP address: %s\n", clientIP);
    printf("Client port number: %d\n", ntohs(servaddr.sin_port));
}
```

A terminal window titled 'cnolasco@calopc: ~/Documents/Ex2-MC833' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command 'sudo ./servidor' being executed. The output is: 'Server port number: 58079', 'Client IP address: 127.0.0.1', and 'Client port number: 50902'. A cursor is visible on the line following the last output.

```
cnolasco@calopc: ~/Documents/Ex2-MC833
cnolasco@calopc:~/Documents/Ex2-MC833$ sudo ./servidor
Server port number: 58079
Client IP address: 127.0.0.1
Client port number: 50902

```

## 7 Questão 8

Sim. O telnet se conecta a um servidor como um cliente, provido de endereço de IP e número da porta.



```
File Edit View Search Terminal Help
cnoiasco@calopc: ~/Documents/Ex2-MC833
cnoiasco@calopc:~/Documents/Ex2-MC833$
cnoiasco@calopc:~/Documents/Ex2-MC833$ sudo ./servidor
Server port number: 36877
Client IP address: 127.0.0.1
Client port number: 38506
```

```
File Edit View Search Terminal Help
cnoiasco@calopc: ~/Documents/Ex2-MC833
cnoiasco@calopc:~/Documents/Ex2-MC833$ telnet 127.0.0.1 36877
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
Mon Oct 26 16:37:43 2020
Connection closed by foreign host.
cnoiasco@calopc:~/Documents/Ex2-MC833$
```