

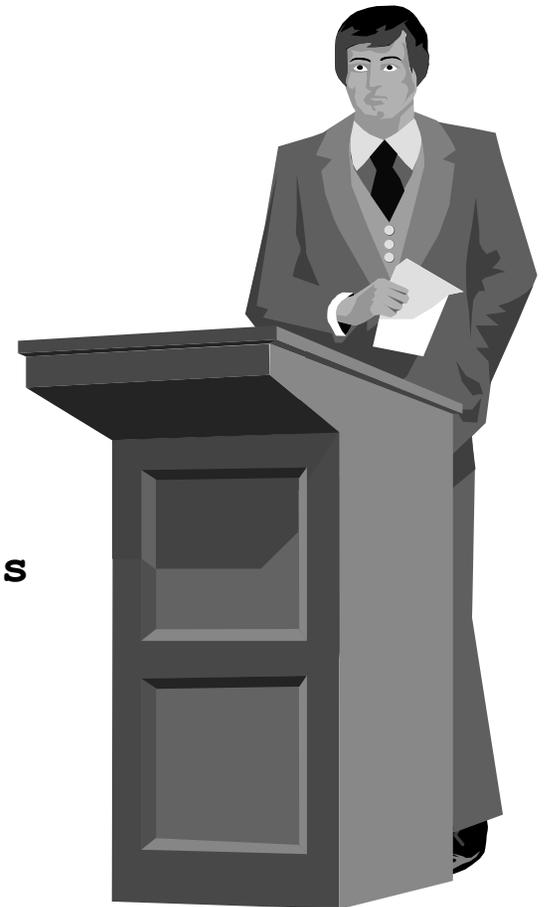
Cliente TCP

Volnys Borges Bernal

`volnys@lsi.usp.br`

Departamento de Sistemas Eletrônicos

Escola Politécnica da USP



Agenda

- **Resumo das chamadas sockets para TCP**

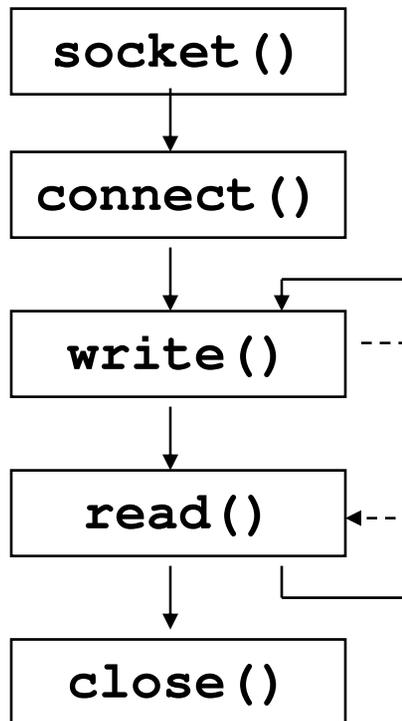
- **Uso das chamadas sockets para TCP:**
 - ❖ Chamada socket()
 - ❖ Chamada connect()
 - ❖ Chamada read()
 - ❖ Chamada write()
 - ❖ Chamada close()

Resumo das chamadas sockets para TCP

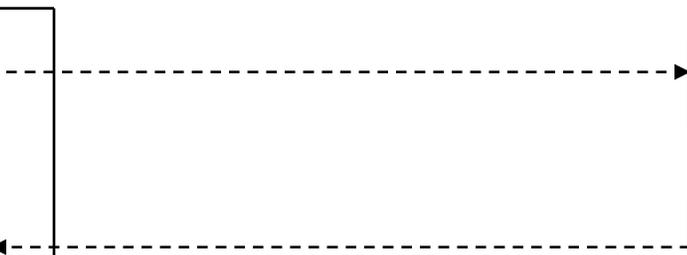
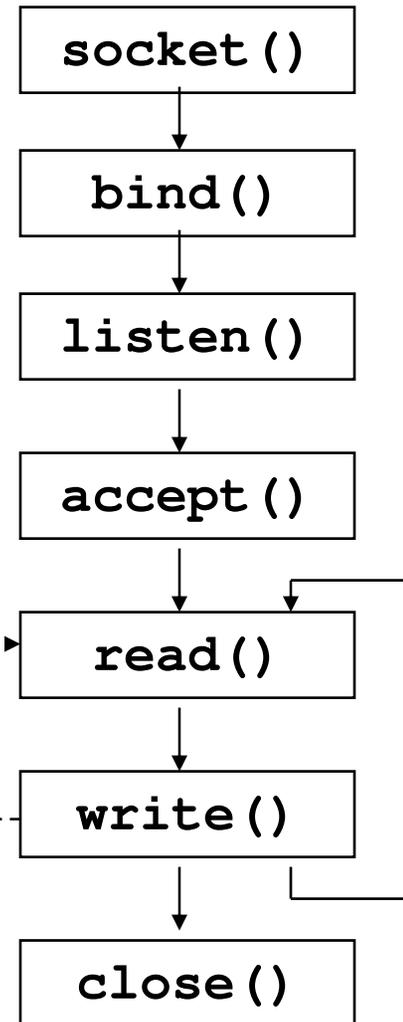


Chamadas sockets para TCP

Lado Cliente



Lado Servidor



Chamada socket()



Chamada socket()

❑ Objetivo

- ❖ Criar um novo socket (plug de comunicação). Aloca estruturas de dados no sistema operacional para suportar a comunicação.

❑ Resultado

- ❖ Retorna o descritor de arquivo (número inteiro).

❑ Sintaxe

```
sd = socket (int domain, int type, int protocol)
```

❑ Observação:

- ❖ Quando um socket é criado, não possui nenhuma informação sobre o parsocket (endereços IPs e portas dos parceiros).

Chamada socket()

□ Sintaxe geral

```
#include <sys/socket.h>
```

```
int socket(int domain, int type, int protocol)
```

*Socket
descriptor*

Para PF_INET use 0

Pilha de protocolos:

- *PF_LOCAL* (file)
- *PF_INET* (IPv4)
- *PF_INET6* (IPv6)
- *PF_X25* (X25)

Tipo da comunicação:

- *SOCK_STREAM* (TCP)
- *SOCK_DGRAM* (UDP)
- *SOCK_RAW* (IP)

Chamada socket()

□ Tipo de serviço

❖ SOCK_STREAM

- Para ser utilizado com o protocolo TCP
- Canal de comunicação full duplex
- Fluxo de bytes sem delimitação
- Chamadas para transmissão e recepção de dados:
 - read(), write() ou send(), recv()

❖ SOCK_DGRAM

- Para ser utilizado com o protocolo UDP
- Datagrama (mensagens)
- Chamadas para transmissão e recepção de dados:
 - send(), sendfrom(), recv() ou recvfrom()

❖ SOCK_RAW

- Permite acesso a protocolos de mais baixo nível
- Datagrama (mensagens)
- Chamadas para transmissão e recepção de dados:
 - send(), recv()

Chamada socket()

- ❑ *Para criar um socket TCP*

```
#include <sys/socket.h>  
sd = socket(AF_INET, SOCK_STREAM, 0);
```

- ❑ *Para criar um socket UDP*

```
#include <sys/socket.h>  
sd = socket(AF_INET, SOCK_DGRAM, 0);
```

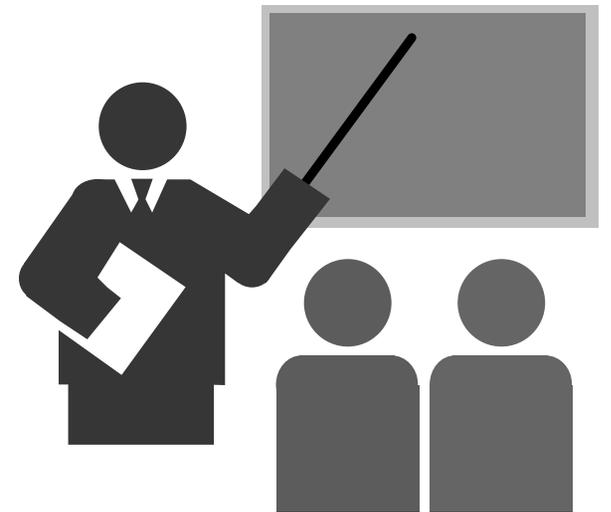
Chamada socket()

❑ Exemplo de criação de socket TCP

```
#include <sys/socket.h>
int sd; // socket descriptor

. . .
sd = socket(PF_INET, SOCK_STREAM, 0); // TCP
if (sd == -1)
{
    perror("Erro na chamada socket");
    exit(1);
}
. . .
```

Chamada connect()



Chamada connect()

□ Objetivo

- ❖ Estabelecer uma sessão de comunicação TCP, UDP ou IP

□ Detalhamento

- ❖ Deve ser utilizado somente no lado cliente
- ❖ UDP:
 - Informa ao sistema operacional o socket address (IP+porta) do parceiro de comunicação
 - Não são enviados datagramas
- ❖ TCP:
 - Informa ao sistema operacional o socket address (IP+porta) do parceiro de comunicação
 - Estabelece a conexão TCP (*3 way handshake*)

Chamada connect()

□ Sintaxe

```
#include <netdb.h>
int connect(int sd,
            struct sockaddr *serveraddr,
            int size)
```

*Socket address (IP +
porta) do parceiro
(servidor)*

*Socket
descriptor*

*Tamanho da estrutura de
endereço (sockaddr_in)*

Chamada connect()

```
#include <netdb.h>

int          status;          //estado da chamada
struct sockaddr_in  serveraddr; //endereço do servidor
...

// define endereço destino
serveraddr.sin_family = AF_INET;
serveraddr.sin_port   = htons(serverport);
status = inet_pton(AF_INET, stringIP, &serveraddr.sin_addr);
if (status <= 0)
    perror("Erro na conversão do endereço IP");

// ativa connect
status = connect( sd,
                 (struct sockaddr *)&serveraddr,
                 sizeof(serveraddr) );
if (status != 0)
    perror("Erro na chamada connect");
```

Chamada connect()

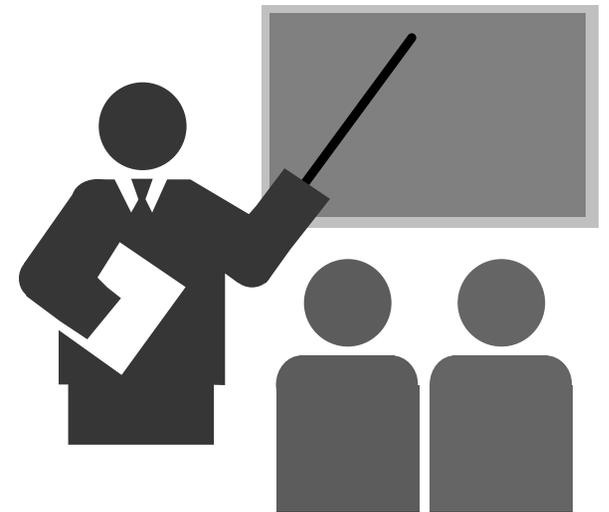
□ Objetivo

- ❖ Estabelecer uma sessão de comunicação TCP, UDP ou IP

□ Detalhamento

- ❖ Deve ser utilizado somente no lado cliente
- ❖ UDP:
 - Informa ao sistema operacional o socket address (IP+porta) do parceiro de comunicação
 - Não são enviados datagramas
- ❖ TCP:
 - Informa ao sistema operacional o socket address (IP+porta) do parceiro de comunicação
 - Estabelece a conexão TCP (*3 way handshake*)

Chamada write()



Chamada write()

❑ **Objetivo**

- ❖ Escrever de dados em um descritor
 - Descritor: descritor sockets, descritor de arquivo, ...
- ❖ Pode ser utilizada no lado cliente ou servidor

❑ **Valor retornado pela função**

- ❖ Positivo: quantidade de bytes escritos
- ❖ -1 : erro

Chamada write()

□ Sintaxe

```
#include <unistd.h>
```

```
int write(int sd, void *txbuffer, int count)
```

*Socket
Descriptor*

*Ponteiro para mensagem
(end. do buffer da mensagem)*

*Tamanho da
mensagem*

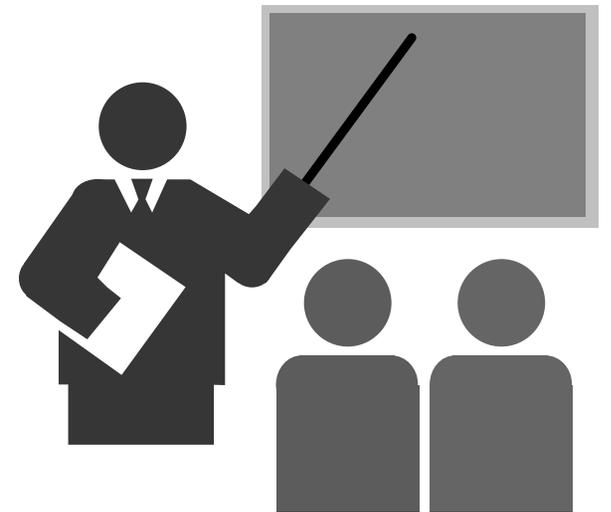
Chamada write()

□ Exemplo:

```
#include <unistd.h>
char txbuffer[80];

. . .
status = write(sd, txbuffer, strlen(txbuffer)+1)
if (status == -1)
    perror("Erro na chamada write");
. . .
```

Chamada read()



Chamada read()

❑ **Objetivo**

- ❖ Ler dados de um descritor
 - Descritor: descritor sockets, descritor de arquivo, ...
- ❖ Pode ser utilizada no lado cliente ou servidor

❑ **Valor retornado pela função**

- ❖ >0: quantidade de bytes lidos
- ❖ 0: end of file
- ❖ -1: erro

Chamada read()

□ Sintaxe:

```
#include <unistd.h>
```

```
int read(int sd, void *rxbuf, int rxbuFSIZE)
```

Socket
Descriptor

Tamanho do
buffer

Ponteiro para o buffer
(end. do buffer de recepção)

Chamada read()

□ Exemplo:

```
char rxbuffer[80];  
  
. . .  
status = read(sd, rxbuffer, sizeof(rxbuffer))  
if (status == -1)  
    perror("Erro na chamada read");  
printf("MSG recebida: %s\n", rxbuffer);  
. . .
```

Chamada close()



Chamada close()

❑ Objetivo

- ❖ Fechar o descritor de arquivos (neste caso, fecha o socket).
- ❖ Se ainda existirem dados para serem transmitidos pelo socket, aguarda por alguns segundos a finalização desta transmissão.

❑ Resultado

- ❖ Fecha o descritor do arquivo.

❑ Sintaxe

```
int close (int sd)
```

Chamada close()

□ Exemplo:

```
int sd; // socket descriptor
. . .

status = close(sd);
if (status == -1)
    perror("Erro na chamada close");
. . .
```

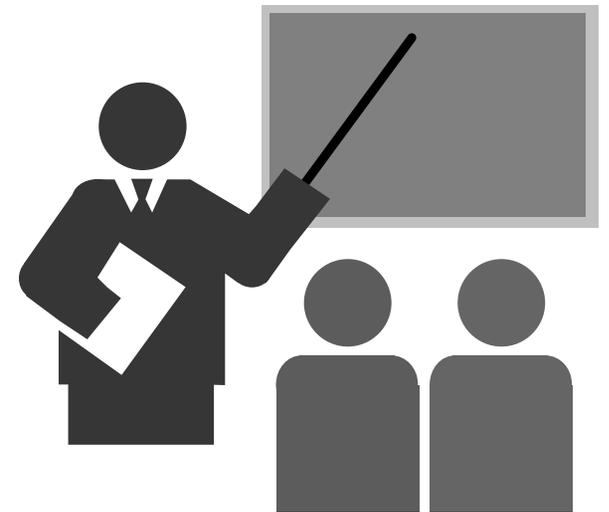
Exercício



Exercício

- (1) **Implemente um cliente para o serviço “echo” utilizando o protocolo TCP.**
-
- **Obs: O serviço TCP echo responde exatamente com a seqüência ASCII enviada.**

Referências Bibliográficas



Referências Bibliográficas

- ❑ **COMMER, DOUGLAS; STEVENS, DAVID**
 - ❖ Internetworking with TCP/IP: volume 3: client-server programming and applications
 - ❖ Prentice Hall
 - ❖ 1993