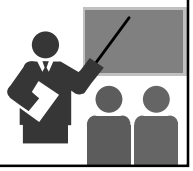
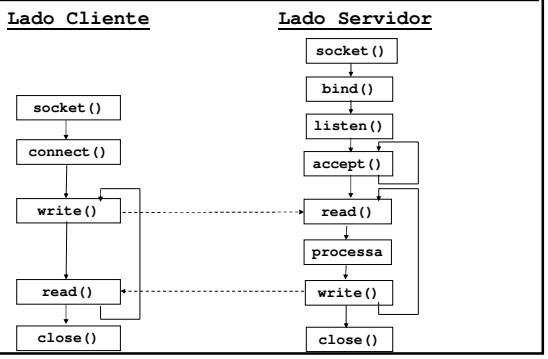
Servidor TCP

Volnys Borges Bernal
volnys@lsi.usp.br
http://www.lsi.usp.br/~volnys

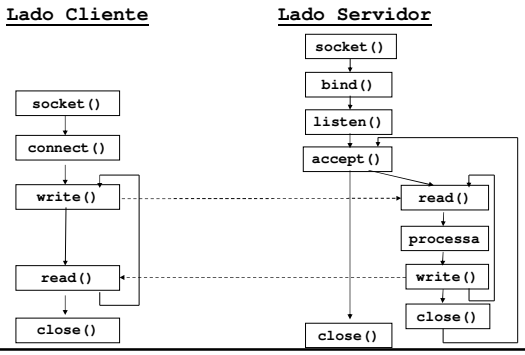
Resumo das Chamadas TCP



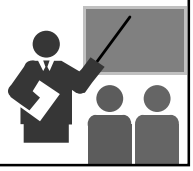
Resumo das Chamadas TCP



Resumo das Chamadas TCP



Chamada bind()



Chamada bind()

- **Objetivo**
  - Permite associar um *socket address* (IP+porta) ao socket
  - Deve ser utilizado no lado servidor
- **Resultado**
  - Retorna –1 no caso de erro

Chamada bind()

- **Chamada bind()**

Endereço IP da interface local.
Pode ser:
- Uma interface específica
- Todas as interfaces locais

```
int  bind( int sockfd,
           struct sockaddr *myaddr,
           socklen_t addrlen)
```

Descritor do socket

Tamanho da estrutura de endereço (sockaddr)

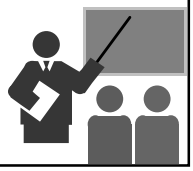
Chamada bind()

- **Exemplo de utilização**

```
struct sockaddr_in mylocal_addr
. . .
mylocal_addr.sin_family      = AF_INET;
mylocal_addr.sin_addr.s_addr = INADDR_ANY;
mylocal_addr.sin_port        = htons(myport);

status = bind(socketdescriptor,
              (struct sockaddr *) &mylocal_addr,
              sizeof(struct sockaddr_in));
if (status == -1)
    perror("Erro na chamada bind");
```


Chamada listen()



Chamada listen()

- **Objetivo**
  - Abrir a porta na qual o servidor ira aguardar conexões TCP
  - As conexões não são aceitas até a ativação de accept()
- **Sintaxe**

```
int listen(int socketdescriptor, int queuelenght)
```

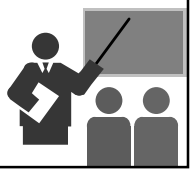
sendo
- `queuelenght`
  - numero máximo de conexões pendentes sem aceite

Chamada listen()

- **Exemplo:**

```
#define QLEN 10
...
status = listen(sd,QLEN);
if (status != 0)
      {
      perror("Erro na chamada listen()");
      exit(1);
      }
```


Chamada accept()



 13

Chamada accept()

- **Objetivo**
  - Aceite de novas conexões TCP
- **Sintaxe**

```
int accept(int sd, struct sockaddr *addr,
                              socklen_t *addrlen)
```

sd	: socket descriptor
addr	: socket address
addrlen	: tamanho da estrutura socket address
valor de retorno	: novo socket descriptor em caso de sucesso valor < 0 em caso de erro

 14

Chamada accept()

- **Exemplo**

```
int                 newsd;
int                 size;
struct sockaddr_in clientaddr;

....
size = sizeof(clientaddr);
newsd = accept( sd,
                (struct sockaddr *) &clientaddr,
                (socklen_t *)       &size);
if (newsd < 0)
        {
        perror("Erro na chamada accept()");
        exit(1);
        }
....
```

 15

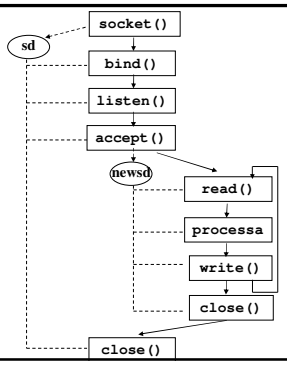
Exercício

1. **Implemente um servidor TCP echo que transforma para maiúsculas.**
   - O programa deve atender a um cliente TCP e, quando receber a mensagem "quit" deve encerrar a conexão com o cliente e terminar o programa
   - Dicas:
     - Utilize a rotina de biblioteca toupper() para converter um caracter minúsculo para maiúsculo.
     - Utilize a rotina de biblioteca strcmp() para comparar a string recebida com "quit".

```
#include <string.h>
If (strcmp(bufferp,"quit")==0)
    /* igual */
```

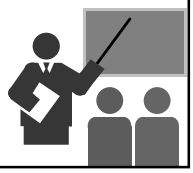
 16

Exercício



 17

Servidor
Não Concorrente x Concorrente

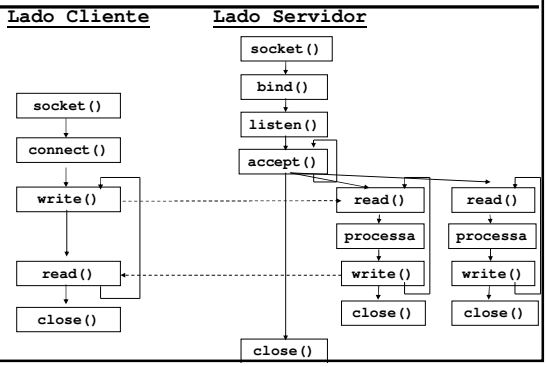


 18

Servidor não concorrente x concorrente

- **Não concorrente**
  - Processa uma requisição por vez
- **Concorrente**
  - Tem capacidade para processar mais que uma requisição simultaneamente
  - Mais complexo
  - Mais difícil de implementar
  - Necessita uma implementação multithreaded

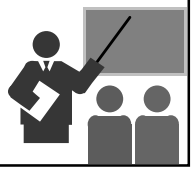
Servidor TCP concorrente



Exercício

- **Fazer um servidor TCP echo concorrente.**
  - Dica: existem diversas formas de implementar o controle dos threads:
  - (a) Utilizar uma tabela de controle de conexões estabelecidas com cada entrada da tabela contendo os seguintes campos: ocupado, estrutura dos threads (thread_t), semáforo para o thread e socket (new socket).
  - (b) Utilizar a estrutura produtor-consumidor. O produtor é o thread principal: produz novas conexões (socket descriptors). O consumidor são os threads de trabalho: consomem conexões (socket descriptors). Um consumidor, ao receber o socket descriptor fica responsável pela interação com o cliente até que a conexão seja encerrada.

Referências Bibliográficas



Referências Bibliográficas

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