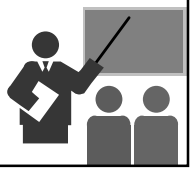
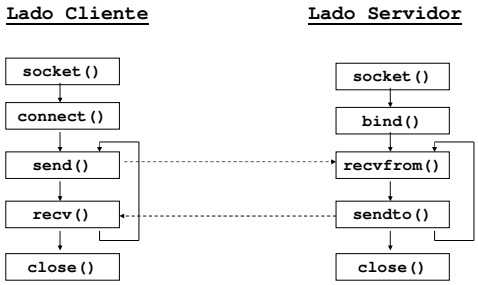
Servidor UDP

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

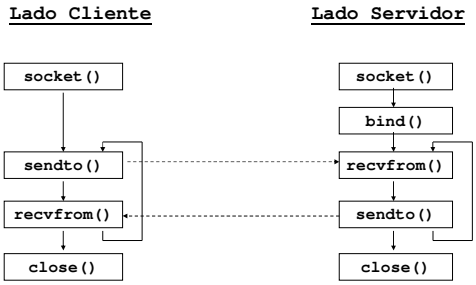
Resumo das Chamadas UDP



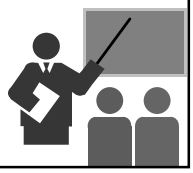
Resumo das Chamadas UDP



Resumo das Chamadas UDP



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()**

```
status =   bind(
               int sd,
               struct sockaddr *myaddr,
               socklen_t addrlen)
```

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

Descritor do socket

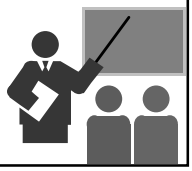
Tamanho da estrutura de endereço (sockaddr)

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");
```

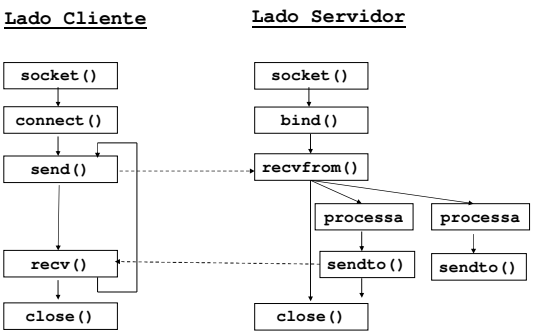

Servidor Não Concorrente x Concorrente



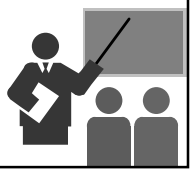
Servidor não concorrente x concorrente

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

Servidor UDP concorrente



Exercício



 13

Exercício

(1) Implemente um servidor para o serviço "UDP echo".

- O serviço "UDP echo" responde exatamente com a seqüência ASCII enviada.

(2) Implemente um servidor para o serviço "UDP daytime TCP"

- O serviço daytime TCP responde com a data e hora do servidor no instante de recebimento de dados TCP.

 14

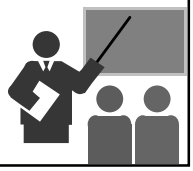
Exercício

(3) Implementar um servidor UDP concorrente que realiza o seguinte processamento:

```
#include <time.h>

void  processar(char* str_in, char* str_out)
    {
    time_t    now_time;
    struct tm now_tm;
    char      str_inicio[40];
    char      str_fim[40];

    now_time = time(NULL);          // Obtém instante (data/hora) corrente
    now_tm   = *localtime(&now);    // Converte para hora local
    // Formata data/hora: "ddd yyyy-mm-dd hh:mm:ss zzz"
    strftime(str_inicio,sizeof(str_inicio),"%a %d/%m/%Y %H:%M:%S %Z",&now_tm);

    sleep(60);
    now_time = time(NULL);          // Obtém instante (data/hora) corrente
    now_tm   = *localtime(&now);    // Converte para hora local
    strftime(str_fim,sizeof(str_inicio),"%a %d/%m/%Y %H:%M:%S %Z", &now_tm);

    sprintf(str_out,"%s: Inicio: %s,  fim: %s ", str_in, str_inicio, str_fim);
    }
```

 15

Referências Bibliográficas



 16

Referências Bibliográficas

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