Introdução ao Laboratório de Soquetes

- O laboratório deve ser feito em linguagem C ou C++

- De preferência em Linux, mas pode ser em Windows.

- Usar a Socket API (Application Programming Interface)

Introduzida no BSD 4.1 - UNIX de Berkeley, 1981
Hoje há para Linux e Windows (winsock)

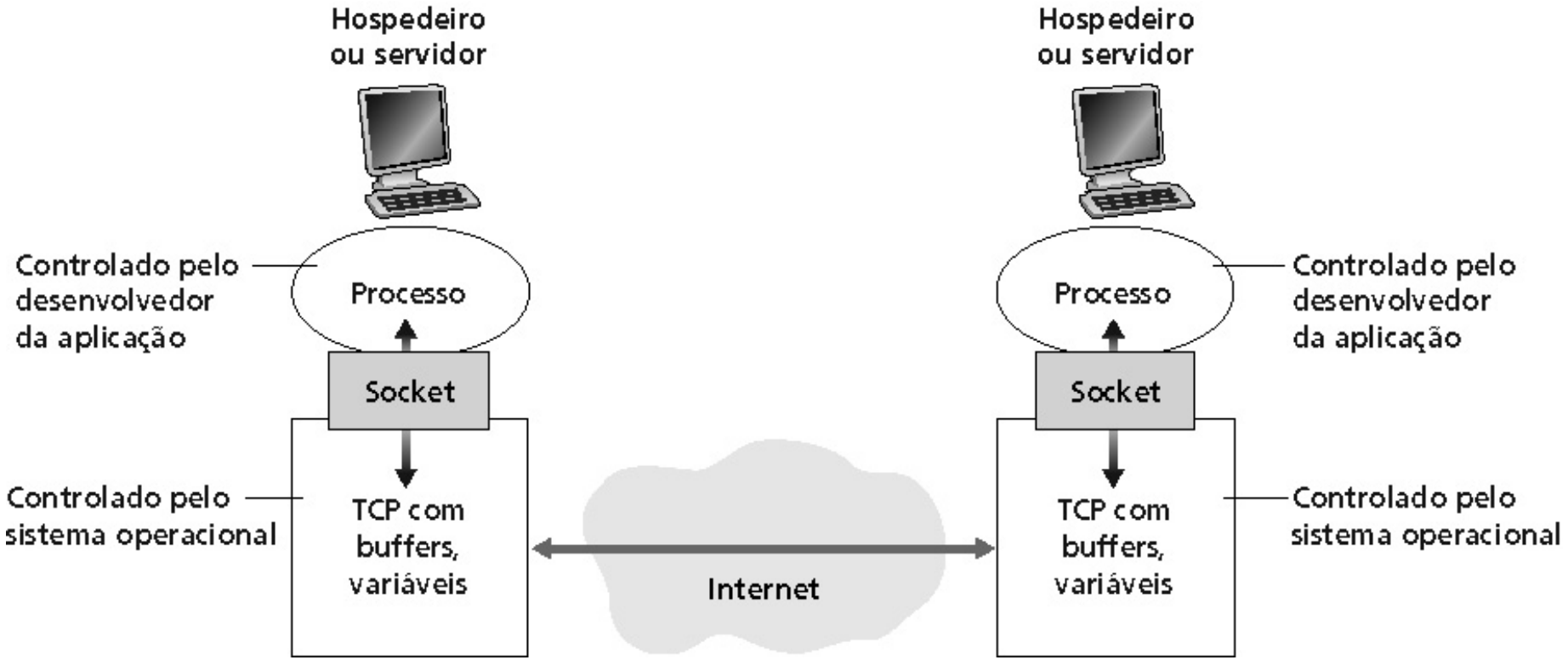
Soquetes (1)

SOCKET

Uma interface **local,** criada por aplicações, ponto final de comunicação no qual os processos de aplicação podem tanto enviar quanto receber mensagens de e para outro processo de aplicação (local ou remoto)

- Explicitamente criados, usados e liberados pelas aplicações
- Paradigma cliente-servidor
- Dois tipos de serviço de transporte via socket API:
  - Datagrama não confiável
  - Confiável, orientado a cadeias de bytes

Soquetes (2)



Serviço TCP: transferência confiável de bytes de um processo para outro

As primitivas de Soquetes para TCP

Retorna descritor de soquete (*s*)

Serv: Vincula IP+Porta a *s*

Serv:

Serv:

Cliente:

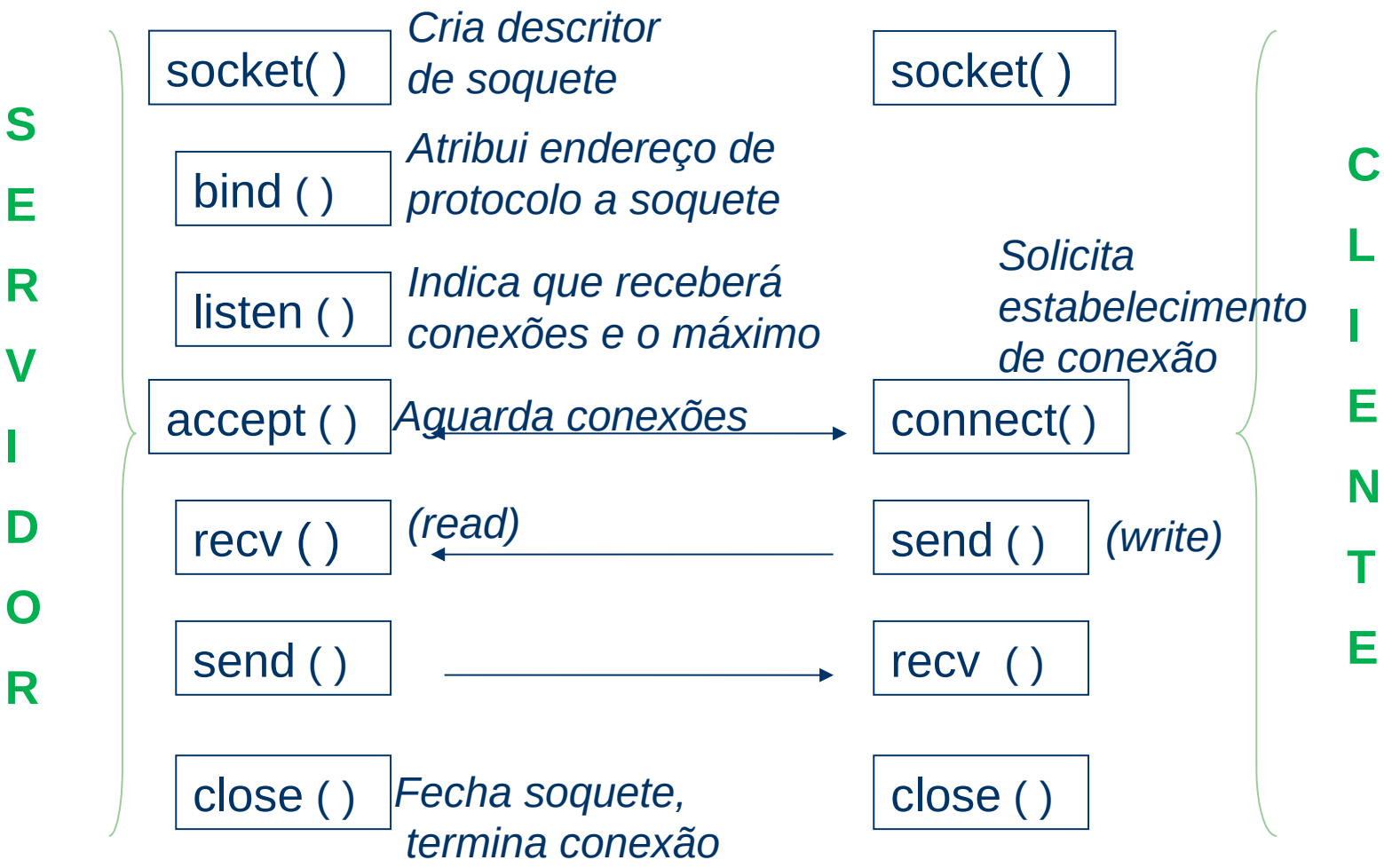
Qualquer:

Qualquer:

Qualquer:

Primitiva	Significado
SOCKET	Criar um novo ponto final de comunicação
BIND	Anexar um endereço local a um soquete
LISTEN	Anunciar a disposição para aceitar conexões; mostrar o tamanho da fila
ACCEPT	Bloquear o responsável pela chamada até uma tentativa de conexão ser recebida
CONNECT	Tentar estabelecer uma conexão ativamente
SEND	Enviar alguns dados através da conexão
RECEIVE	Receber alguns dados da conexão
CLOSE	Encerrar a conexão

Pseudo-código



Informações complementares (1)

- Little Endian versus Big Endian: maneiras de armazenar um número na memória. Little Endian: armazena os bytes de menor ordem em primeiro. Comum na arquitetura Intel. Ex: 0x12345678 seria armazenado como (0x78 0x56 0x34 0x12)

  Big Endian: 0x12345678 seria armazenado como (0x12 0x34 0x56 0x78). Comum nas arquiteturas RISC. É o formato da rede.

- Função para padronizar a transmissão:

Htons: host to network – unsigned short int to Bigendian.

Htonl: host to network – unsigned long int to Bigendian.

Informações complementares (2)

- É montada uma estrutura de dados (*struct sockaddr_in channel*) para especificar um endereço local ou remoto de ponto de extremidade para conectar o soquete:

channel.sin_addr.s_addr=htonl(INADDR_ANY)

Server Address

"This allowed your program to work without knowing the IP address of the machine it was running on, or, in the case of a machine with multiple network interfaces, it allowed your server to receive packets destined to any of the interfaces".

- Define de que versão virá a família de endereços:

channel.sin_family = AF_INET

AF_INET: Address Family for Internet Sockets is IPv4

Código do Cliente (1)

Solicita arquivo do servidor. Chamada client <url> <file> >f

```
/* Esta página contém um programa cliente que pode solicitar um arquivo do*/
/* programa servidor na próxima página. O servidor responde enviando o arquivo inteiro.*/


#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345                /* arbitrário, mas cliente e servidor devem combinar */
#define BUF_SIZE 4096                    /* tamanho do bloco de transferência */

int main(int argc, char **argv)
{
  int c, s, bytes;
  char buf[BUF_SIZE];                    /* buffer para arquivo de entrada */
  struct hostent *h;                     /* informações sobre servidor */
  struct sockaddr_in channel;            /* mantém endereço IP */
```

Código do Cliente (2)

```
if (argc != 3) fatal("Usage: client server-name file-name");
h = gethostbyname(argv[1]);                    /* pesquisa endereço IP do host */
if (!h) fatal("gethostbyname failed");

s = socket(PF_INET,SOCK_STREAM,IPPROTO_TCP);
if (s <0) fatal("socket");
memset(&channel, 0, sizeof(channel));
channel.sin_family= AF_INET;
memcpy(&channel.sin_addr.s_addr, h->h_addr,h->h_length);
channel.sin_port= htons(SERVER_PORT);

c = connect(s, (struct sockaddr *) &channel, sizeof(channel));
if (c < 0) fatal("connect failed");
```

Código do Cliente (3)

```
  /* Conexão agora estabelecida. Envia nome do arquivo com byte 0 no final. */
  write(s, argv[2], strlen(argv[2])+1);

  /* Captura o arquivo e o escreve na saída padrão. */
  while (1) {
    bytes = read(s, buf, BUF_SIZE);          /* lê do soquete */
    if (bytes <= 0) exit(0);                 /* verifica final de arquivo */
    write(1, buf, bytes);                    /* escreve na saída padrão */
  }
}

fatal(char *string)
{
  printf("%s\n", string);
  exit(1);
}
```

Código do Servidor (1)

```
#include <sys/types.h>                          /* Este é o código do servidor */
#include <sys/fcntl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345                       /* arbitrário, mas cliente e servidor devem combinar */

#define BUF_SIZE 4096                           /* tamanho do bloco de transferência */
#define QUEUE_SIZE 10

int main(int argc, char *argv[])
{
  int s, b, l, fd, sa, bytes, on = 1;
  char buf[BUF_SIZE];                           /* buffer para arquivo de saída */
  struct sockaddr_in channel;                   /* mantém endereço IP */
```

Código do Servidor (2)

```
/* Monta estrutura de endereços para vincular ao soquete. */
memset(&channel, 0, sizeof(channel));                    /* canal zero */
channel.sin_family = AF_INET;
channel.sin_addr.s_addr = htonl(INADDR_ANY);
channel.sin_port = htons(SERVER_PORT);

/* Abertura passiva. Espera a conexão. */
s = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);    /* cria soquete */
if (s < 0) fatal("socket failed");
setsockopt(s, SOL_SOCKET,SO_REUSEADDR,(char *) &on, sizeof(on));

b = bind(s, (struct sockaddr *) &channel, sizeof(channel));
if (b < 0) fatal("bind failed");

l = listen(s, QUEUE_SIZE);                               /* especifica tamanho da fila */
if (l < 0) fatal("listen failed");
```

Código do Servidor (3)

```
/* O soquete agora está preparado e vinculado. Espera conexão e a processa. */
while (1) {
  sa = accept(s, 0, 0);                          /* bloqueia solicitação de conexão */
  if (sa < 0) fatal("accept failed");

  read(sa, buf, BUF_SIZE);                       /* lê nome do arquivo do soquete */

  /* Captura e retorna o arquivo. */
  fd = open(buf, O_RDONLY);                      /* abre arquivo para ser enviado de volta */
  if (fd < 0) fatal("open failed");

  while (1) {
    bytes = read(fd, buf, BUF_SIZE);             /* lê do arquivo */
    if (bytes <= 0) break;                       /* verifica se é final do arquivo */
    write(sa, buf, bytes);                       /* grava bytes no soquete */
  }
  close(fd);                                     /* fecha arquivo */
  close(sa);                                     /* fecha conexão */
 }
}
```