Aula Prática 2007 - 03

Sockets
Comunicação entre processos no Linux

Disciplina: INF01151
Prof. Dr. Cláudio Fernando Resin Geyer
Monitor: Eder Stone Fontoura

Sockets

- **Objetivo**
  - Apresentar na prática a comunicação entre processos usando sockets
- **Roteiro**
  - Sockets
  - Sockets Linux
  - TCP Socket
  - UDP Socket

Sockets

- **Um socket é uma abstração que indica a extremidade final de um canal de comunicação**
- **Na prática, um socket é uma interface de comunicação bidirecional entre processos**
- **Em outras palavras, um socket é por onde um processo envia ou recebe dados de outros processos**
- **Associados a um socket existem diversos modelos e primitivas para a realização da comunicação entre processos distintos**

Sockets Linux

- **Linux utiliza sockets como abstração geral para vários tipos de entrada e saída**
- **No que diz respeito a redes, o Linux implementa a interface conhecida como Berkeley socket interface**
  - Uma biblioteca C lançada em 1983 como parte do BSD 4.2
  - Define um conjunto de funções que permitem a troca de dados entre processos
  - Considerada o padrão de facto para uma interface de programação de rede

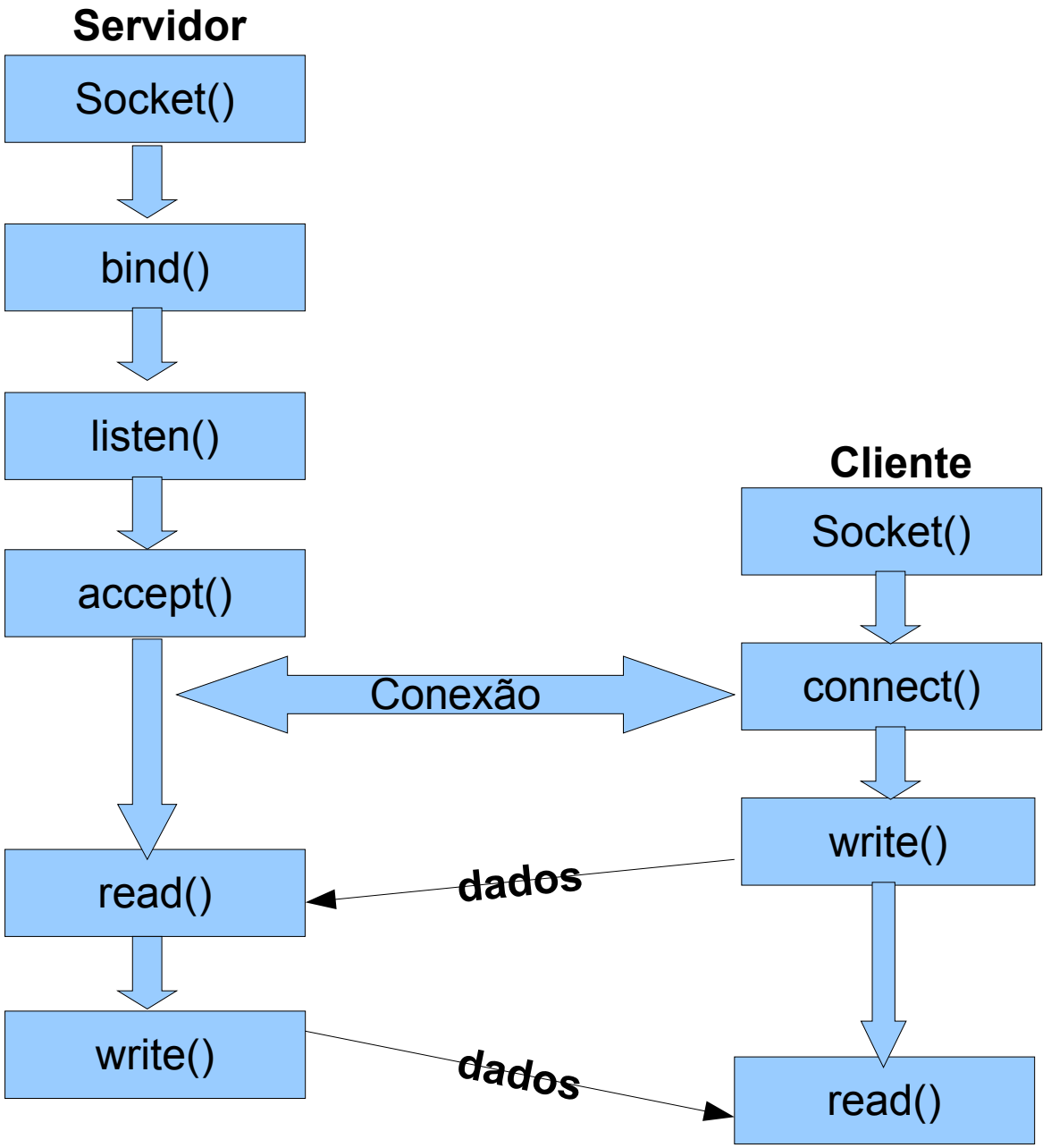
Sockets Linux

- **A interface édefinida basicamente nos seguintes arquivos**
  - <sys/socket.h>: definições das funções e tipos de dados básicos
  - <sys/types.h>: definições de tipos de dados
  - <netinet/in.h>: definições de tipos de dados relacionados a endereços
- **Ela oferece chamadas de sistema e funções para**
  - Criação e associação de endereço do socket
  - Conexão e desconexão de um socket
  - Escuta e aceite de conexões
  - Envio e recebimento de dados
  - Manipulação de endereços

Sockets Linux

- **Tipos de domínio**
  - Apenas os que serão usados. Existem alguns outros que podem ser encontrados na documentação
  - Unix (A**F_UNIX**) – dois processos que compartilham um mesmo sistema de arquivos se comunicam
  - Internet (A**F_INET**) – dois processos executando em quaisquer dois hosts na internet se comunicam
- **Tipos de sockets**
  - Stream (**SOCK_STREAM**) – comunicação bidirecional confiável (entrega de pacotes garantida e em ordem correta) – protocolo default TCP
  - Datagram (**SOCK_DGRAM**) – comunicação bidirecional não-confiável(não existe garantia de entrega nem de ordem de entrega de pacotes) - protocolo default UDP

Sockets – SOCK_STREAM (TCP)



Sockets - SOCK_STREAM(TCP)

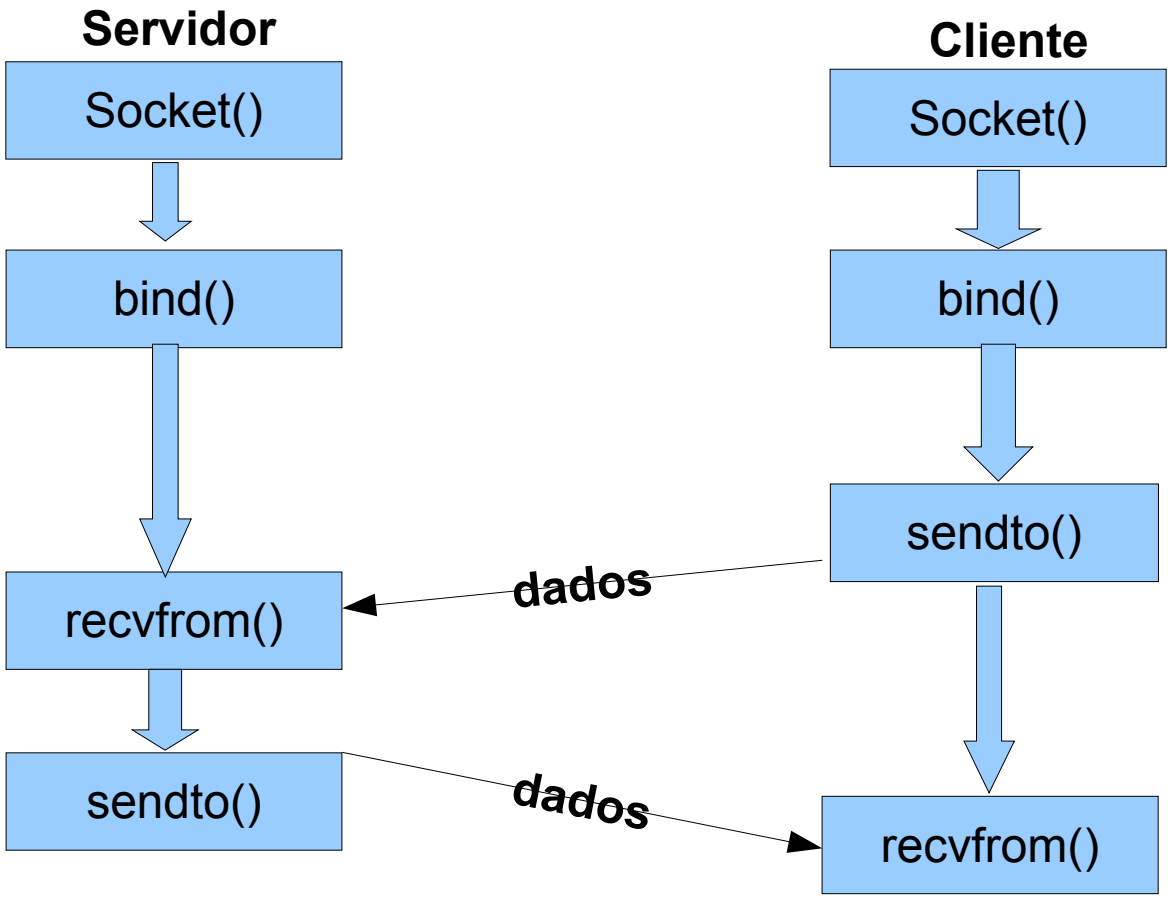
- **Servidor (Exemplo)**

```
//Criacao do socket no servidor
serversocket = socket(AF_INET,SOCK_STREAM,0);
//Ligando o endereco do servidor ao socket
bind(serversocket, (struct sockaddr *)&server_addr, sizeof(struct sockaddr);
 //Indica que o socket deve "escutar"
listen(serversocket, 1);
printf("Servidor escutando na porta: %d .\n", PORTNUMBER);
 //Servidor aguardando por conexao
connectionsocket = accept(serversocket, (struct sockaddr *)&client_addr,
    &socketsize);
 //Lendo mensagem enviada pelo cliente
tamr = read(connectionsocket, buffer, 100);
```

- **Cliente (Exemplo)**

```
clientSock = socket(PF_INET, SOCK_STREAM, 0);
//Efetua conexao com o sevidor
conn = connect(clientSock, (struct sockaddr *) &s, sizeof(s));
tams = write(clientSock,buffer,strlen(buffer));
```

Sockets – SOCK_DGRAM (UDP)



Sockets Linux – System Calls

- **int socket(int domain, int type, int protocol) -** Criar socket
- **int bind(int sockfd, struct sockaddr *my_addr, int addrlen) -** Ligar o socket a um endereço
- **int listen(int sockfd, int backlog) -** Ficar “escutando” requisições
- **int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen) -** Aceita/aguarda um pedido de conexão. Gera um descritor especifico para cada conexão
- **int connect(int sockfd, struct sockaddr *addr, socklen_t *addrlen) –** Solicitar uma conexão
- **int send(int sockfd, constvoid*msg, int len, int flags) int sendto(int sockfd, constvoid*msg, int len, unsignedint flags, conststructsockaddr*to, socklen_ttolen) – mensagens em sockets tipo SOCK_STREAM pode-se usar send e sendto. Com o tipo SOCK_DGRAM somente sendto**
- **int recv(int sockfd, void*buf, int len, unsignedint flags) int recvfrom(int sockfd, void*buf, int len, unsignedint flags, structsockaddr*from, int *fromlen) - mensagens em sockets tipo SOCK_STREAM pode-se usar recv e recvfrom. Com o tipo SOCK_DGRAM somente recvfrom**
- **close(sockfd) – fechar o socket**

Exercícios

1. **Aponte as principais semelhanças e as principais diferenças entre as 2 formas de comunicação apresentadas a partir do código e do que se observou na execução dos exemplos em relação aos seguintes itens**
   1. Criação do socket
   2. Nomeação e endereçamento dos processos
   3. Modelo de comunicação (passos para realização de todo o ciclo de comunicação)
   4. Chamadas usadas para troca de dados
2. **No socket SOCK_STREAM(TCP) o servidor sabe quando o cliente se desconecta? E no socket SOCK_DGRAM(UDP) ?**

Exercícios

3. **Modifique o código do cliente/servidor UDP para que tenha mais de um cliente na mesma máquina (use Threads)**
4. **Modifique o código do exemplo cliente/servidor TCP para tornar o servidor multithreaded, ou seja, a cada cliente que o servidor aceitar uma thread deve ser criada pelo servidor para atender aquele cliente**