SISTEMAS OPERACIONAIS

Comunicação entre Processos

Andreza Leite
andreza.leite@univasf.edu.br

Plano de Aula

- Introdução a Comunicação entre Processos
- Pipes
- Sockets
- RMI
- Corba

Comunicação entre Processos

- Em ambientes de multipramação é comum que processos se comuniquem uns com os outros.
- Muitos SOs disponibilizam mecanismos para IPC (*Interprocess Comunication*).
  - Habilitam um editor de texto a enviar um doc para o spooler de uma impressora ou um navegador web a recuperar dados de um servidor remoto.
- Essencial para processos que devem coordenar(sincronizar) atividades para alcançar meta comum

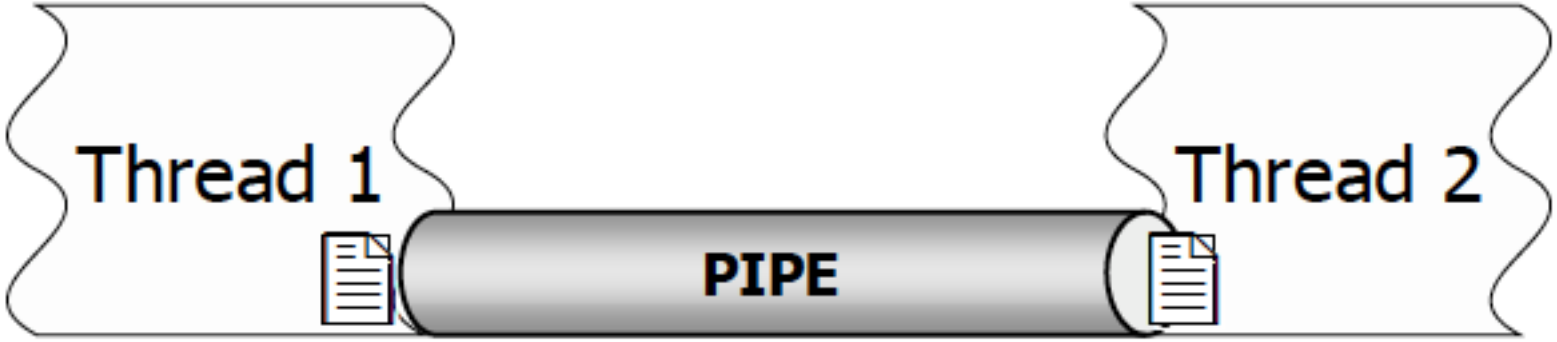
IPC – Toca de mensagens

- Com os SDs cresceu o interesse na comunicação baseada em mensagens.
- Mensagens podem ser passadas em uma direção por vez
  - Processo age como emissor e receptor
  - Troca de mensagens pode ser bidirecional
  - Fazem chamadas como:

```
send(processoReceptor, mensagem)
receive(processoEmissor, mensagem)
```

Pipes

- Implementação popular da troca de mensagens.
- Uma região da memória serve como buffer com acesso sincronizado pelo SO.
  - Um processo escritor escreve os dados. SO pausa sua execução permitindo que o leitor leia os dados.
  - O leitor vai fazendo a leitura dos dados (esvaziando o buffer). SO pausa a execução do leitor e permite que o escritor volte a escrever.



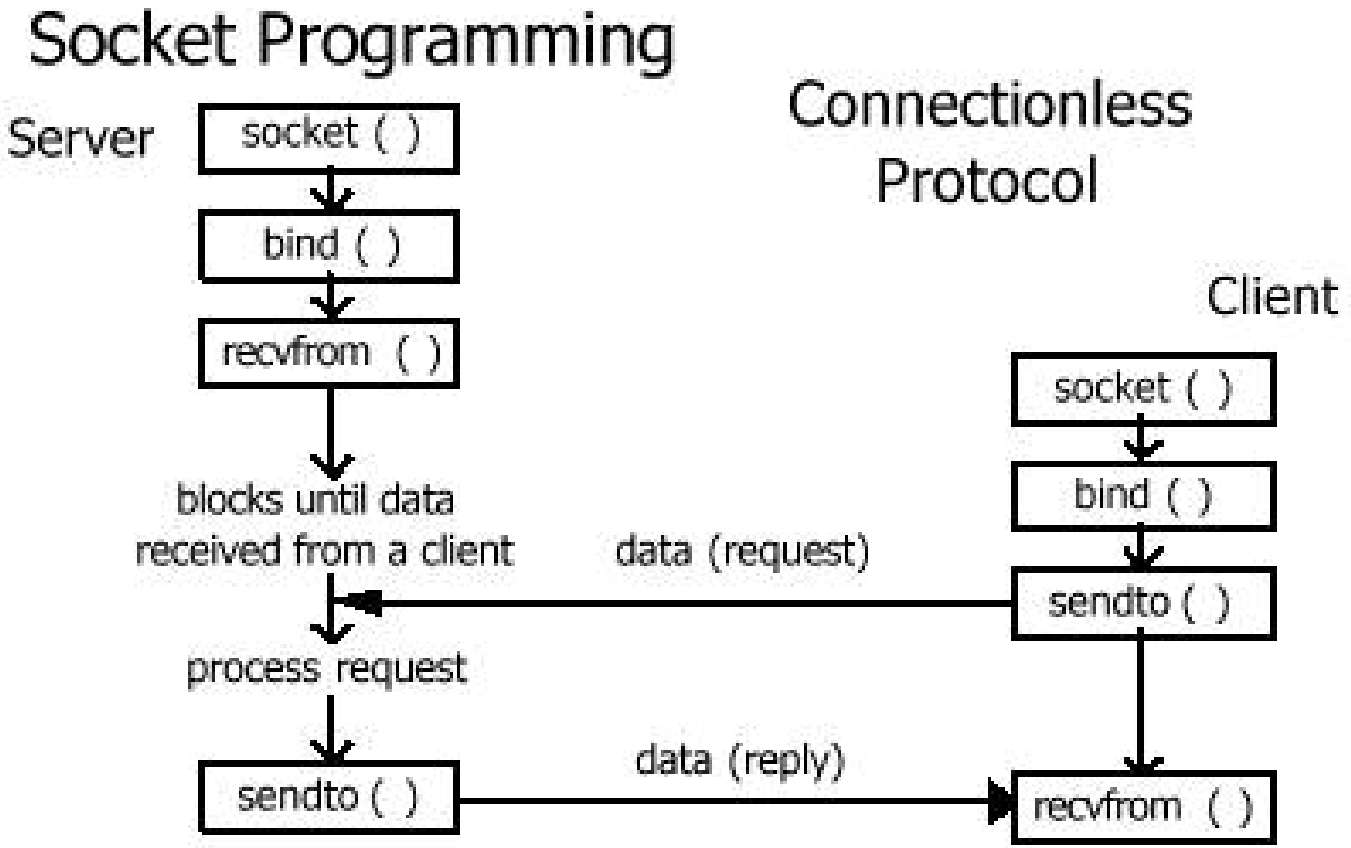
Socket

- Componente de software disponibilizado pelo SO para permitir que um programa receba corretamente pacotes de dados direcionados a ele.
  - Um programa cria um Socket Server e o ativa para poder receber solicitações.
  - Este Socket Server é caracterizado por um endereço
  - Outros programas que conheçam esse endereço podem se comunicar com o programa servidor.

Socket

- Um exemplo passos envolvidos no estabelecimento de um socket:
  - Criar um socket com a chamada ao sistema socket().
  - Amarrar o socket a um endereço usando a chamada ao sistema bind().
    - Na Internet, um endereço consiste num par IP/porta.
  - Enviar e receber dados com chamada ao sistema recvfrom() e sendto()

Socket

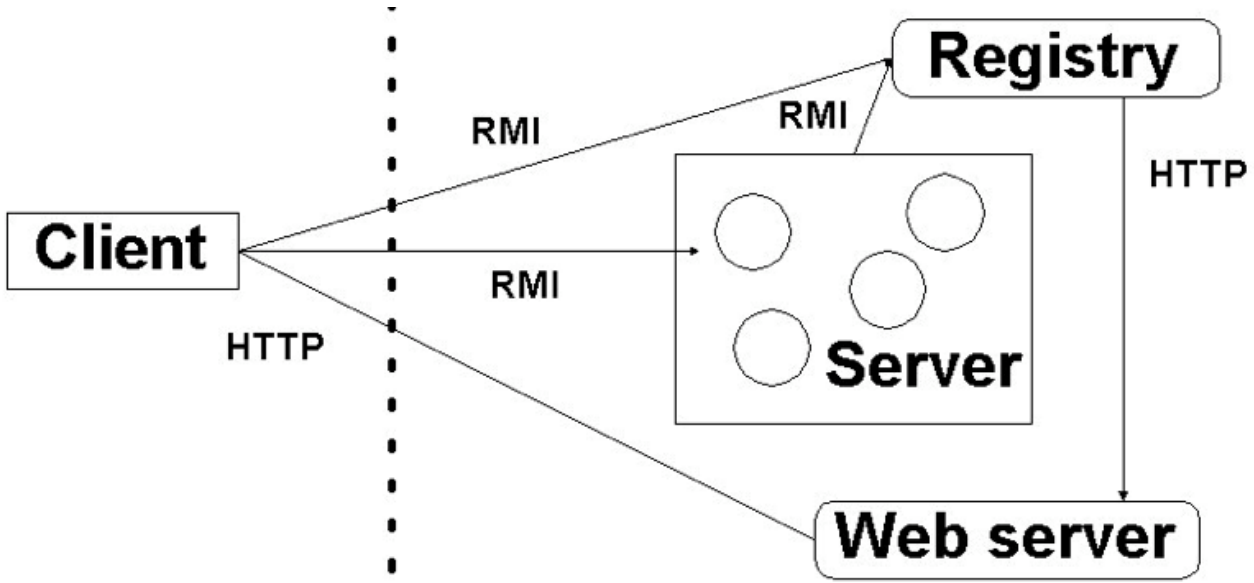

Socket Programming
Connectionless Protocol
Server
socket ( )
bind ( )
recvfrom ( )
blocks until data received from a client
process request
sendto ( )
Client
socket ( )
bind ( )
sendto ( )
recvfrom ( )
data (request)
data (reply)

Java RMI (*Remote Method Invocation*)

- RMI é uma interface de programação que permite a execução de chamadas remotas no estilo RPC.
- Uma das abordagens da plataforma Java para prover as funcionalidades de uma plataforma de objetos distribuídos.
- O funcionamento de RMI consiste basicamente em dois programas, segundo a arquitetura cliente-servidor.
  - O servidor instancia objetos remotos, o referencia com um nome e faz um "BIND" dele numa porta, onde este objeto espera por clientes que invoquem seus métodos.
  - Já o cliente referencia remotamente um ou mais métodos de um objeto remoto.
- RMI fornece os mecanismos para que a comunicação entre cliente e servidor seja possível.

Java RMI

- Uma aplicação pode usar dois mecanismos para obter referências de objetos remotos:
  - Ela pode registrar o objeto remoto com a ferramenta de nomes do RMI, "rmiregistry",
  - Ou passar e retornar referências aos objetos remotos como parte de sua operação normal.



CORBA

- CORBA (*Common Object Request Broker Architecture*) é a arquitetura padrão criada pelo *Object Management Group* para estabelecer e simplificar a troca de dados entre sistemas distribuídos heterogêneos (hardware e software).

- CORBA atua de modo que os objetos (componentes dos softwares) possam se comunicar de forma transparente ao usuário, mesmo que para isso seja necessário interoperar com outro software, em outro SO e em outra ferramenta de desenvolvimento.

Corba

- Define o ORB (*Object Request Broker*) como um módulo intermediário entre cliente e objeto
  - Responsável em aceitar a requisição do cliente, enviá-la para o objeto competente e, assim que disponível a resposta, entregá-la para o cliente.