Sistemas Operacionais

Processos

Mecanismos Mais Elaborados de Comunicação e Sincronização entre Processos

- A troca de mensagens é um mecanismo de comunicação e sincronização que exige do S.O., tanto a sincronização quanto a comunicação entre os processos.
- Os mecanismos já considerados exigem do S.O. somente a sincronização, deixando para o programador a comunicação de mensagens através da memória compartilhada.
- Os mecanismos estudados até agora asseguram a exclusão mútua, mas não garantem um controle sobre as operações desempenhadas sobre o recurso.

Processos

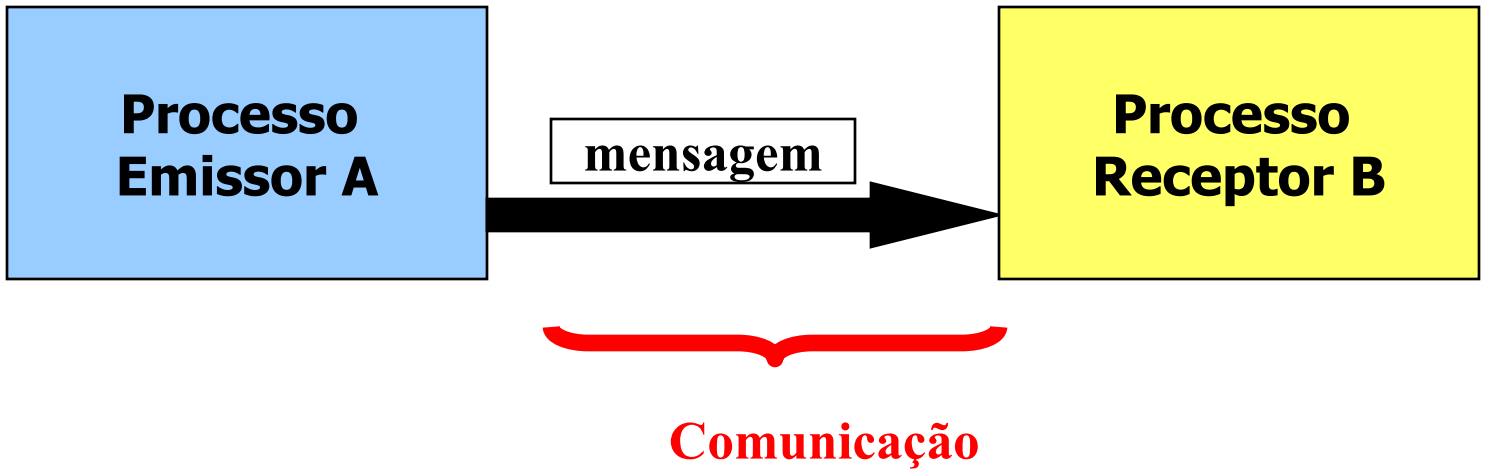
- O uso de troca de mensagens para a manipulação de um recurso compartilhado assegura a exclusão mútua, e impõe restrições nas operações a serem desempenhadas sobre ele.

- Os mecanismos já considerados exigem do S.O. somente a sincronização, deixando para o programador a comunicação de mensagens através da memória compartilhada.

- Esquema de troca de mensagens: os processos enviam e recebem mensagens, em vez de ler e escrever em variáveis compartilhadas.

Processos

- **A sincronização entre processos: é garantida pela restrição de que uma mensagem só poderá ser recebida depois de ter sido enviada.**
- **A transferência de dados de um processo para outro, após ter sido realizada a sincronização, estabelece a comunicação.**



Processos

Primitivas de Troca de Mensagens

- De forma genérica, uma mensagem será <u>enviada</u> quando um processo executar o seguinte comando:

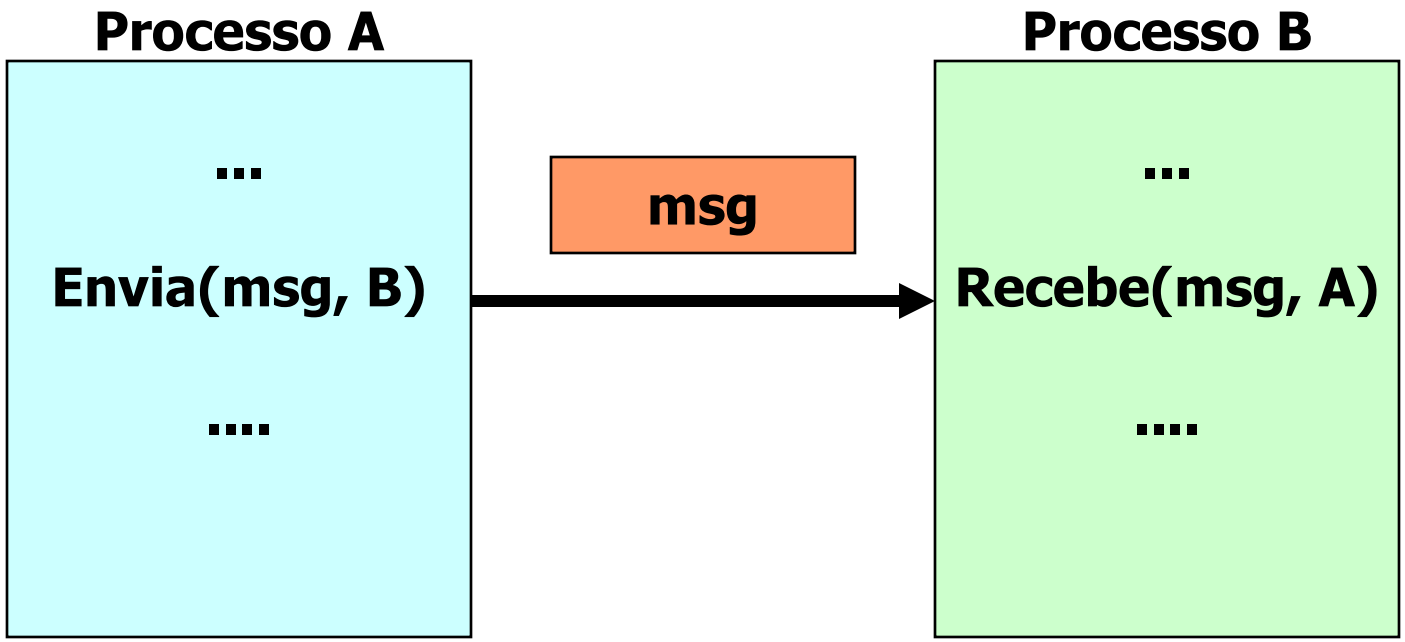
Envia (mensagem, processo_receptor)
ou
Send(message, receiver)

- Uma mensagem será <u>recebida</u> quando um processo executar o seguinte comando:

Recebe(mensagem, processo_emissor)
ou
Receive(message, sender)

Processos

Primitivas de Troca de Mensagens



Processos

Primitivas de Troca de Mensagens

- **As primitivas podem ser de dois tipos:**

  - **Bloqueantes: quando o processo que a executar ficar bloqueado até que a operação seja bem sucedida (ou seja, quando ocorrer a entrega efetiva da mensagem ao processo destino, no caso da emissão, ou o recebimento da mensagem pelo processo destino, no caso de recepção).**

  - **Não bloqueantes: quando o processo que executar a primitiva, continuar sua execução normal, independentemente da entrega ou do recebimento efetivo da mensagem pelo processo destino.**

Processos

Exemplo de Comunicação usando Troca de Mensagens

```
Program emissor_receptor;
Type msg=...;
Var mensagem : msg;
Begin  /* inicio do programa principal */
  Cobegin /* inicio dos processos concorrentes */
    Begin /* processo emissor – E */
     repeat
       ...;
       produz uma mensagem;
       Send(mensagem, R);
     until  false
    End;

    Begin /* processo receptor – R */
     repeat
       Receive(mensagem, E);
       Consome a  mensagem;
        ...;
       until  false
    End
  Coend
End.
```

Processos

Os sistemas de troca de mensagens possuem alguns problemas e estudos de projetos interessantes, principalmente quando os processos comunicantes estão em máquinas diferentes, conectadas por uma rede de comunicação. Os principais são:

- **Perda de mensagens**
- **Perda de reconhecimento**
- **Nomeação de Processos**
- **Autenticação**
- **Estudos de Projeto para quando emissor e receptor estiverem na mesma máquina**

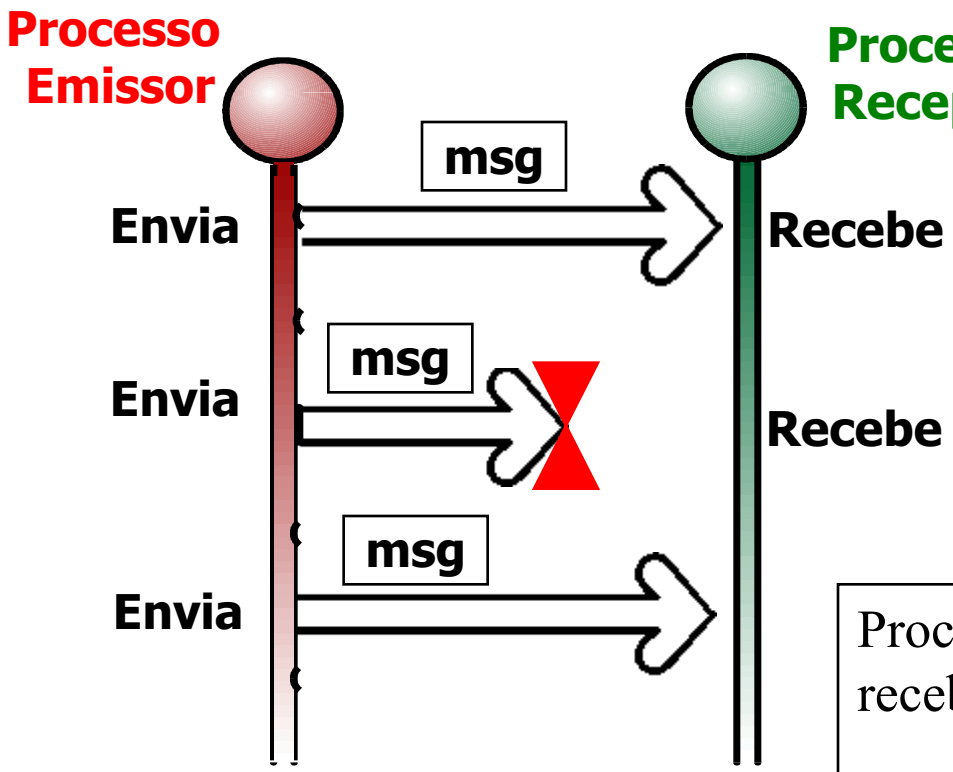
Processos

- **Perda de Mensagens**

uma solução é que o receptor, ao receber uma nova mensagem, envie uma mensagem especial de reconhecimento (ACK). Se o emissor não receber um ACK a tempo, deve retransmitir a mensagem.

Processos

Perda de Mensagens: o problema



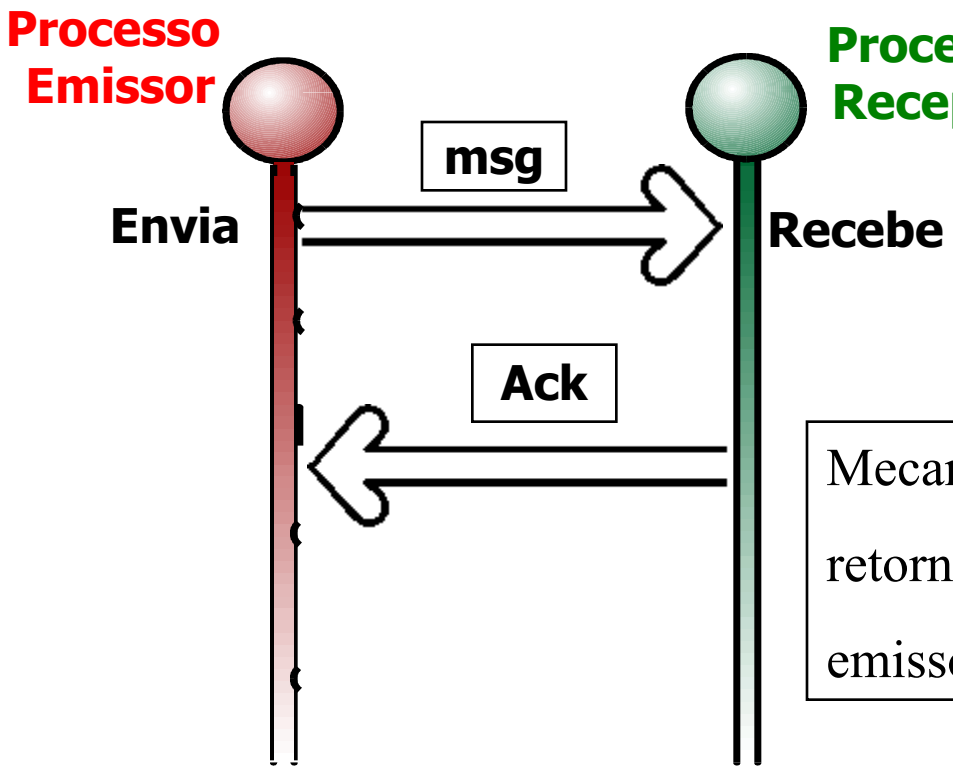
Processo receptor recebeu mensagem com sucesso

Processo receptor não recebeu a nova mensagem, devido a algum problema

Processo emissor não sabe que receptor não recebeu a mensagem anterior e envia uma nova mensagem

Processos

Perda de Mensagens: uma solução



Processo receptor recebeu mensagem com sucesso

Mecanismo de troca de mensagens retorna um ACK para a máquina do emissor

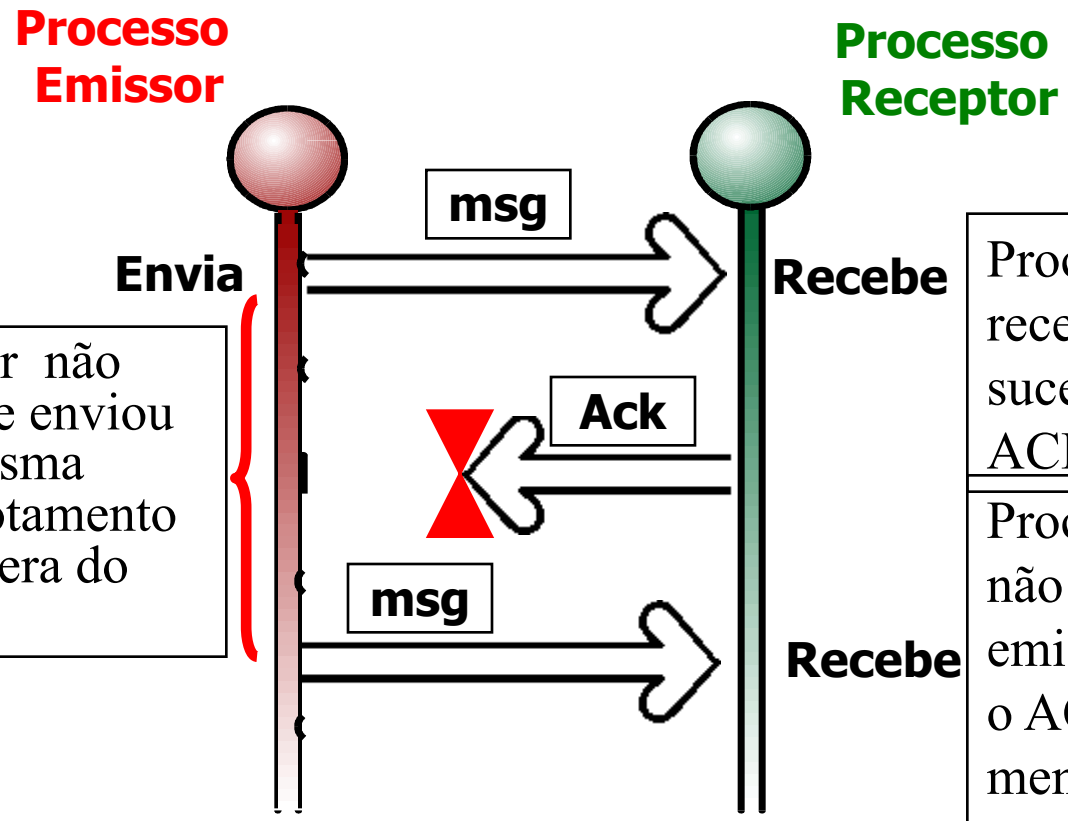
Processos

- **Perda de Reconhecimento (ACK)**

causa o problema de se receber mensagens idênticas.
Uma das soluções é a numeração de mensagens.

Processos

Perda de Reconhecimento: o problema



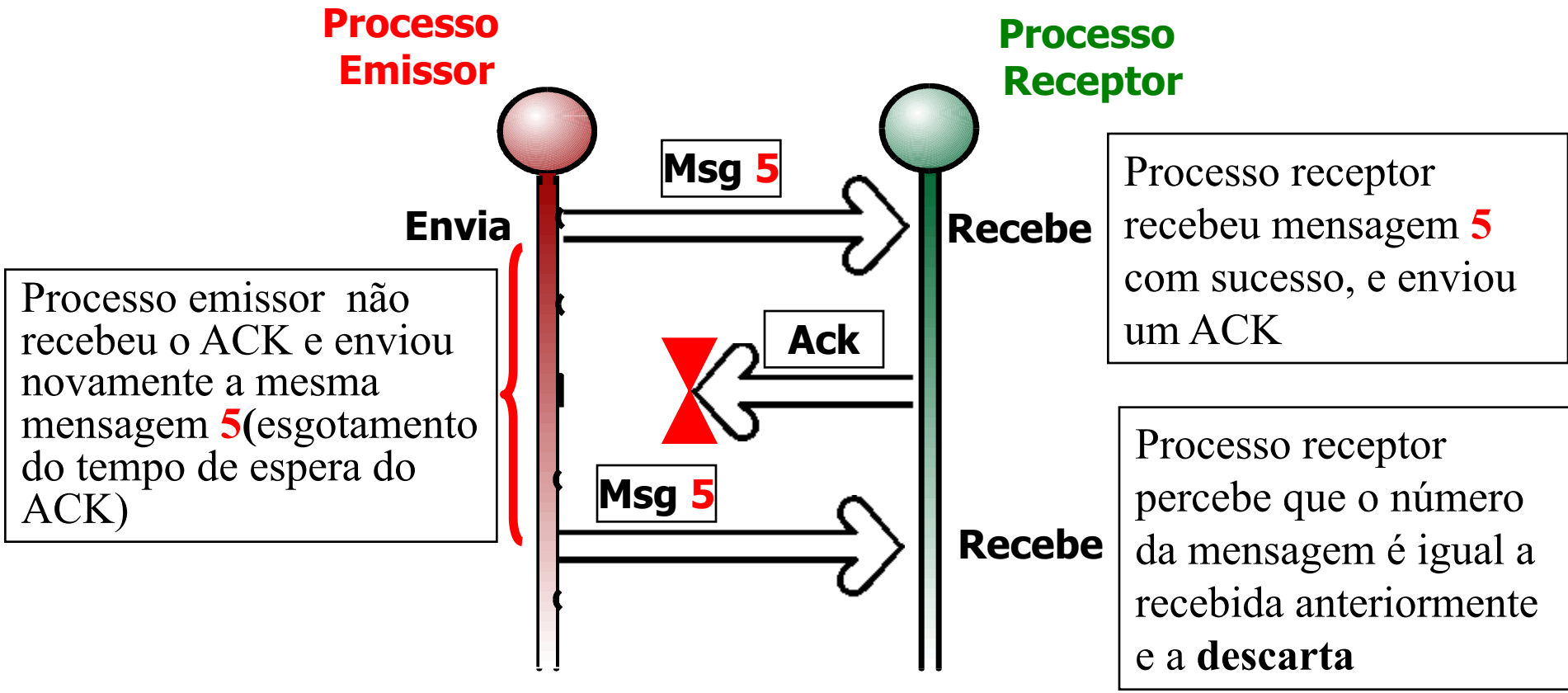
Processo emissor não recebeu o ACK e enviou novamente a mesma mensagem (esgotamento do tempo de espera do ACK)

Processo receptor recebeu mensagem com sucesso, e enviou um ACK

Processo receptor não sabe que o emissor não recebeu o ACK e considera a mensagem recebida como nova (duplicação)

Processos

Perda de Reconhecimento: uma solução é numerar as mensagens



Processos

- **Nomeação de Processos**

os processos devem ser nomeados de maneira única, para que o nome do processo especificado no Send ou Receive não seja ambíguo.

Ex: processo@máquina (normalmente existe uma autoridade central que nomeia as máquinas).
Quando o número de máquinas é muito grande: processo@máquina.domínio.

Processos

- **Autenticação**

permitir a comunicação e acessos apenas dos usuários autorizados. Uma solução é criptografar as mensagens com uma chave conhecida apenas por usuários autorizados.

Processos

- **Estudo de projeto para quando emissor e receptor estão na mesma máquina**

para aumento do desempenho, pensa-se em registradores especializados para a troca de mensagens.

Processos

Combinação de Primitivas

existem quatro maneiras de se combinar as primitivas de troca de mensagens:

Envia Bloqueante – Recebe Bloqueante → **síncrono**

Envia Bloqueante – Recebe Não Bloqueante } **Semi síncrono**

Envia Não Bloqueante – Recebe Bloqueante } **Semi síncrono**

Envia Não Bloqueante – Recebe Não Bloqueante → **Assíncrono**

Processos

Mecanismos de Comunicação Síncronos

Existem três mecanismos de comunicação síncronos mais importantes:

- **Rendez-vous**
- **Rendez-vous Estendido**
- **Chamada Remota de Procedimento**

Processos

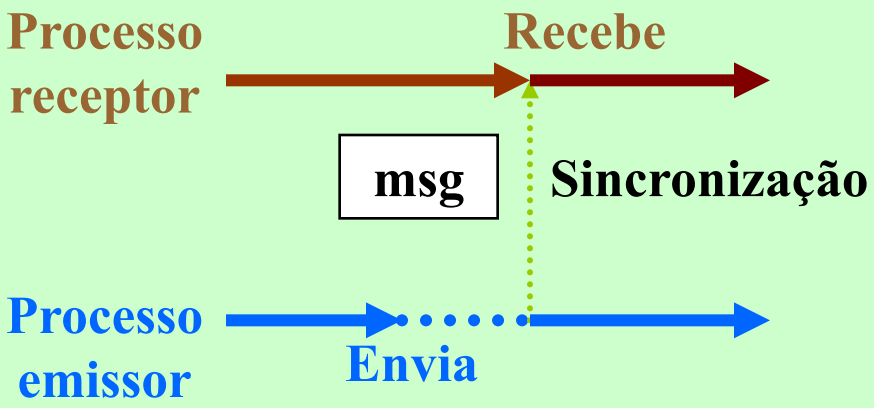
- **Rendez-vous**

é obtido através de primitivas Envia e Recebe bloqueantes colocadas em processos distintos; a execução destas primitivas em tempos diferentes, fará com que o processo que executar a primitiva antes do outro fique bloqueado até que ocorra a sincronização entre os dois processos, e a consecutiva transferência da mensagem; em seguida, ambos os processos continuarão seu andamento em paralelo.
Ex.: linguagem CSP.

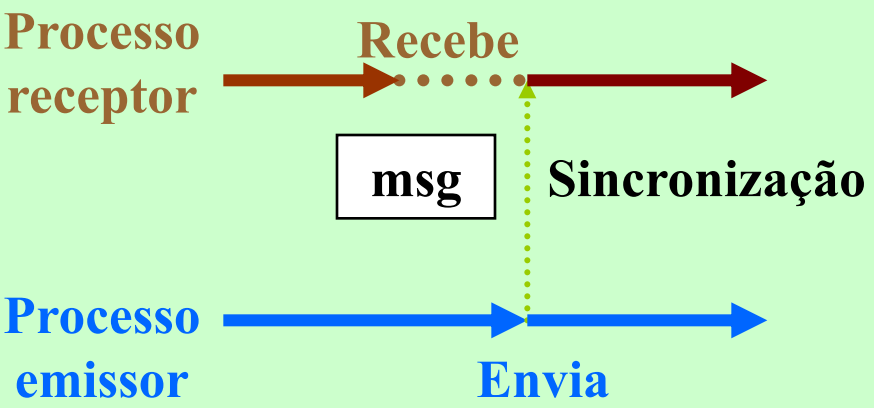
Processos

Rendez-vous

Emissor Antecipado



Receptor Antecipado



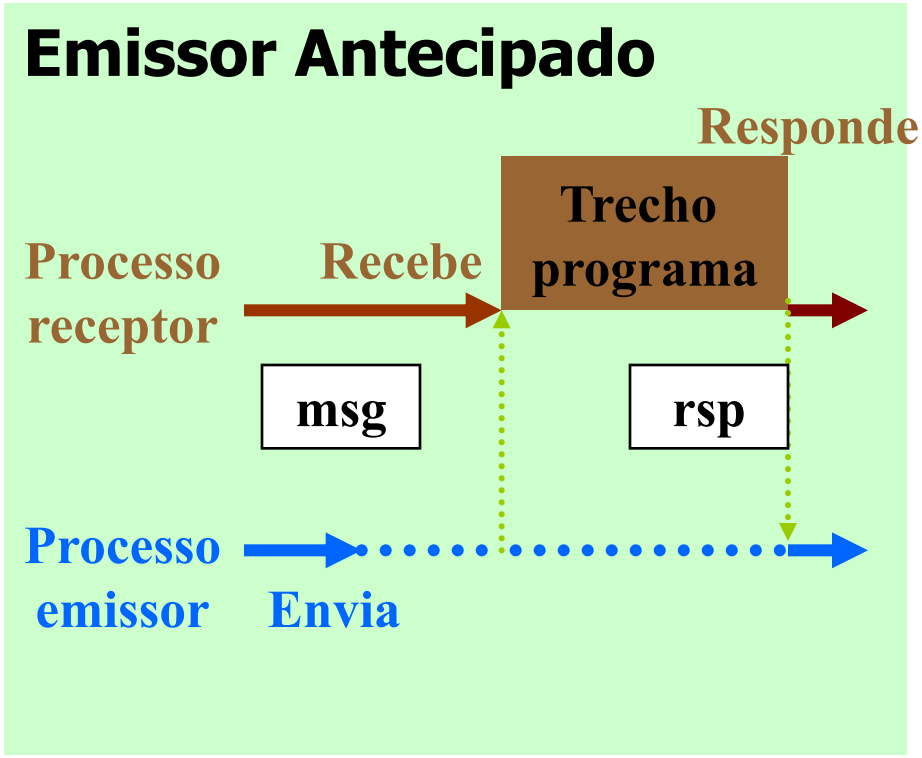
Processos

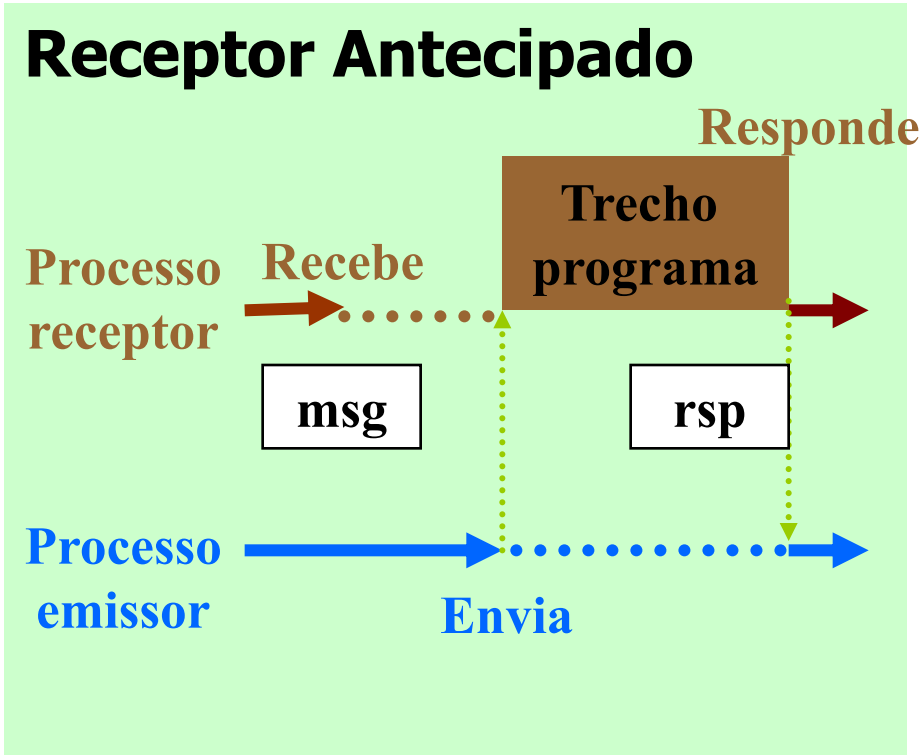
- **Rendez-vous Estendido**

caracteriza-se por apresentar uma estrutura de comunicação onde um processo consegue comandar a execução de um trecho de programa previamente estabelecido, pertencente a outro processo, envolvendo sincronização e, eventualmente, troca de mensagem.
Ex: linguagem ADA.

Processos

Rendez-vous Estendido





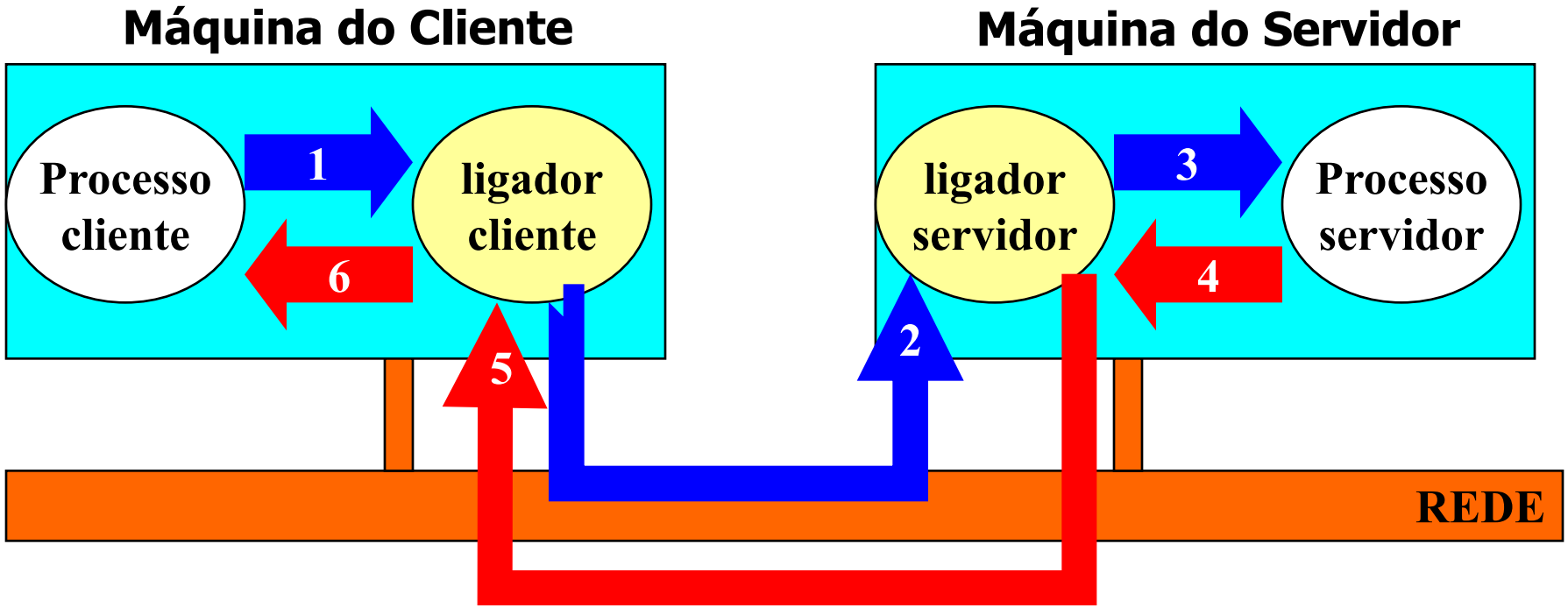
Processos

- **Chamada Remota de Procedimento (RPC - Remote Procedure Call)**

apresenta uma estrutura de comunicação na qual um processo pode comandar a execução de um procedimento situado em outro processador. O processo chamador deverá ficar bloqueado até que o procedimento chamado termine. Tanto a chamada quanto o retorno podem envolver troca de mensagem, conduzindo parâmetros.
Ex: linguagem DP.

Processos

Chamada Remota de Procedimento



(1) e (3) são chamadas de procedimento comuns
(2) e (5) são mensagens
(4) e (6) são retornos de procedimento comuns

Processos

Vantagem da Chamada Remota de Procedimento

- **Cliente e servidor não precisam saber que as mensagens são utilizadas.**

Eles as veem como chamadas de procedimento locais.

Processos

Problemas da Chamada Remota de Procedimento

- **Dificuldade da passagem de parâmetros por referência:** por exemplo, se as máquinas servidora e cliente possuem diferentes representações de informação (necessidade de conversão e desconversão).
- **Diferenças de arquitetura:** por exemplo, as máquinas podem diferir no armazenamento de palavras.
- **Falhas semânticas:** por exemplo, o servidor pára de funcionar quando executava uma RPC. O que dizer ao cliente? Se disser que houve falha e o servidor terminou a chamada logo antes de falhar, o cliente pode pensar que falhou antes de executar a chamada. Ele pode tentar novamente, o que pode não ser desejável. Principais abordagens: "no mínimo uma vez", "exatamente uma vez" e "talvez uma vez".

Processos

Outros Mecanismos de Comunicação Baseados na Troca de mensagens

Existem diversos mecanismos atuais baseados na troca de mensagens, além dos discutidos. Os mais representativos são:

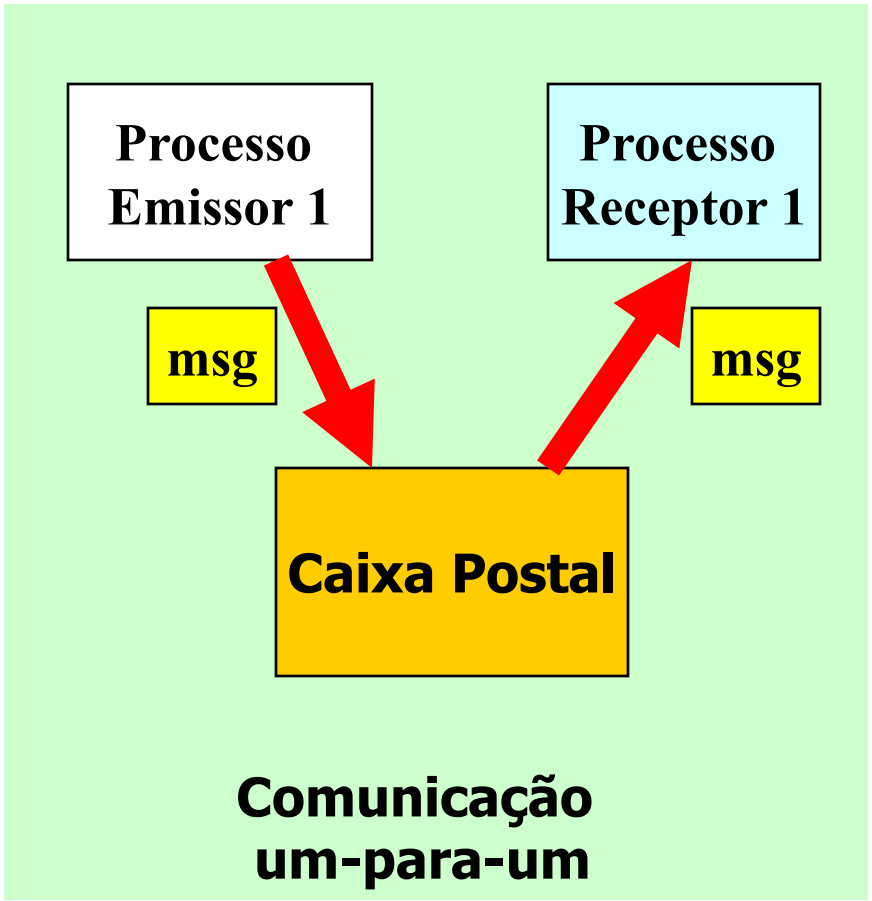
- **Caixas Postais**
- **Portos**

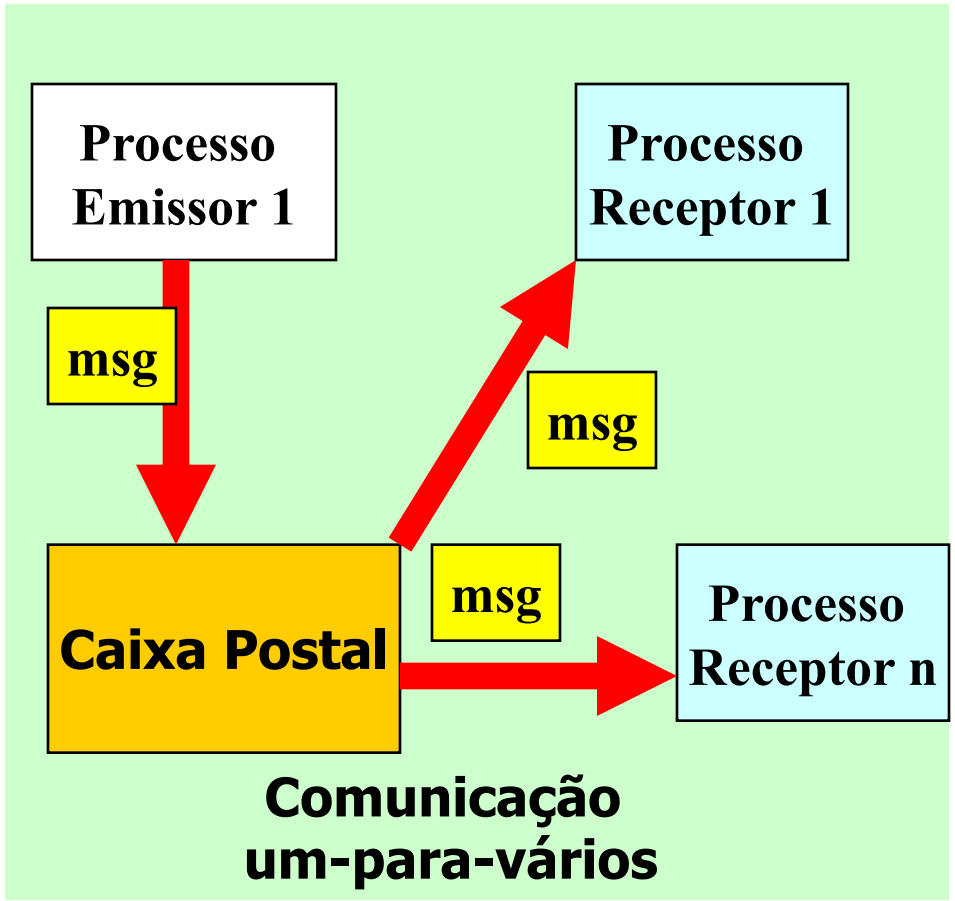
Processos

- **Caixas Postais (Mailboxes)**
- **São filas de mensagens não associadas, a princípio, com nenhum processo.**
- **Os processos podem enviar e receber mensagens das caixas postais, tornando o mecanismo adequado para comunicar diversos remetentes a diversos receptores.**
- **Uma restrição ao uso de caixas postais é a sua necessidade de envio de duas mensagens para comunicar o remetente com o receptor: uma do remetente à caixa postal, e outra da caixa postal para o receptor.**

Processos

Caixas Postais





Processos

- **Portos (Ports)**
  - **consiste num elemento do sistema que permite a comunicação entre conjuntos de processos através do mascaramento de um de seus identificadores. Cada porto é como uma caixa postal, porém com um dono, que será o processo que o criar.**

Processos

Portos

Cria_porto(S, saída, msg)

conecta_porto(S, E)

envia_porto(S, msg)

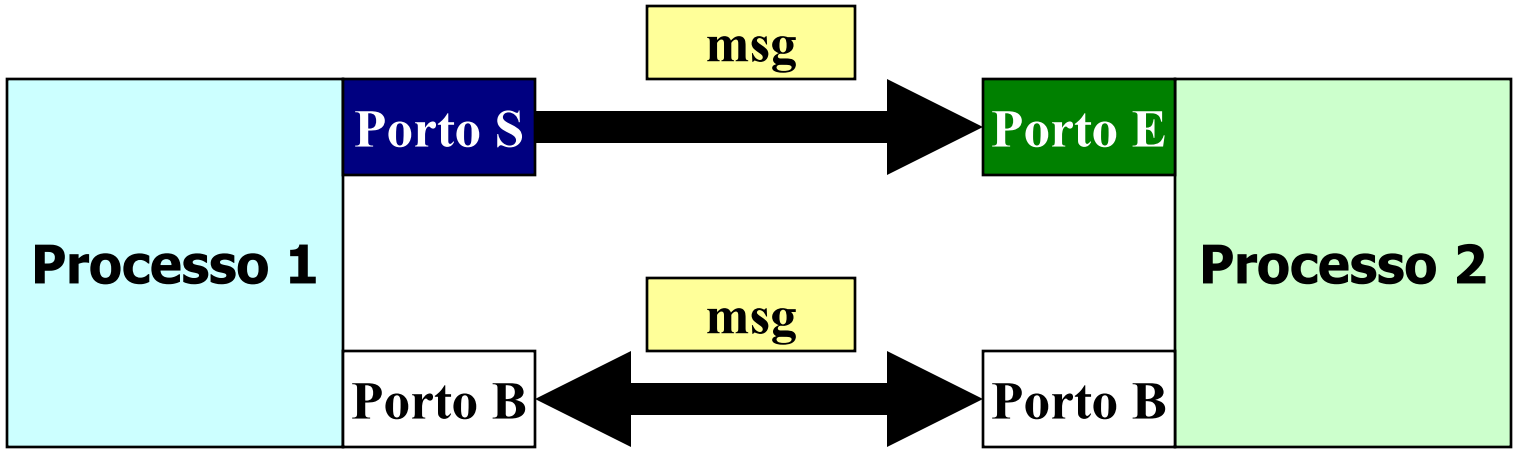
desconecta_porto(S, E)

destruir_porto(S)

Cria_porto(E, entrada, msg)

recebe_porto(E, msg)

destruir_porto(E)



Processos

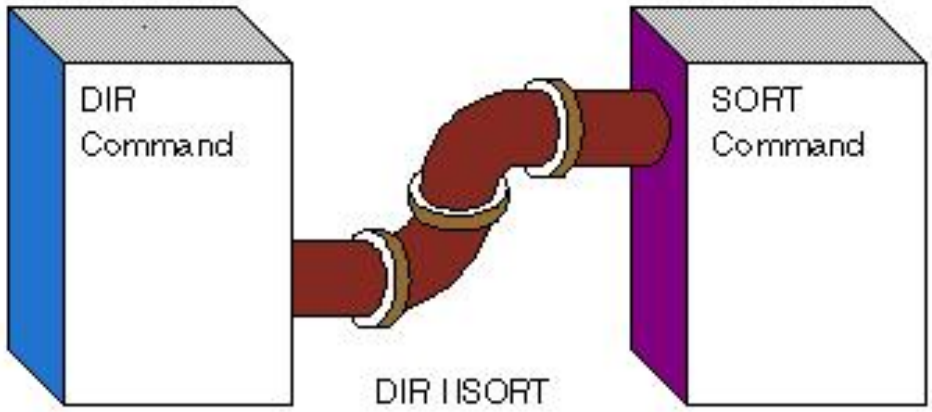
Outras Características dos Portos

- A criação e a interligação de portos e caixas postais pode ser feita de maneira dinâmica. A necessidade de enfileiramento das mensagens enviadas, torna necessário o uso de "buffers", para o armazenamento intermediário.
- A comunicação entre processos locais ou remotos, num sistema estruturado com portos, será feita pela execução de primitivas síncronas (ou assíncronas) do tipo envia e recebe.

Processos

Comunicação de Processos no UNIX: PIPES

- **Pipes: – usados no sistema operacional UNIX para permitir a comunicação entre processos. Um pipe é um modo de conectar a saída de um processo com a entrada de outro processo, sem o uso de arquivos temporários. Um pipeline é uma conexão de dois ou mais programas ou processos através de pipes.**



Processos

PIPES

Exemplos:

sem uso de pipes (usando arquivos temporários)

```
$ ls > temp
$ sort < temp
```

com uso de pipes

```
$ ls | sort
```