Unix Windows NT Netware MacOS DOS/VS Vax/VMS
Linux Solaris HP/UX AIX Mach
Chorus

Programação Concorrente

por Troca de Mensagens:

I- Introdução

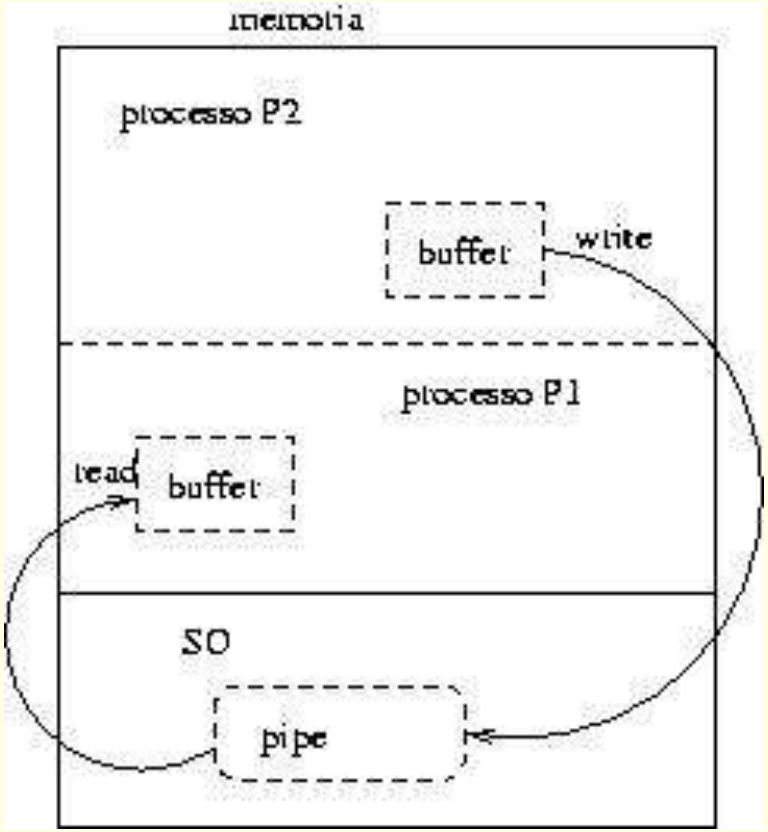
Resumo de aulas anteriores…

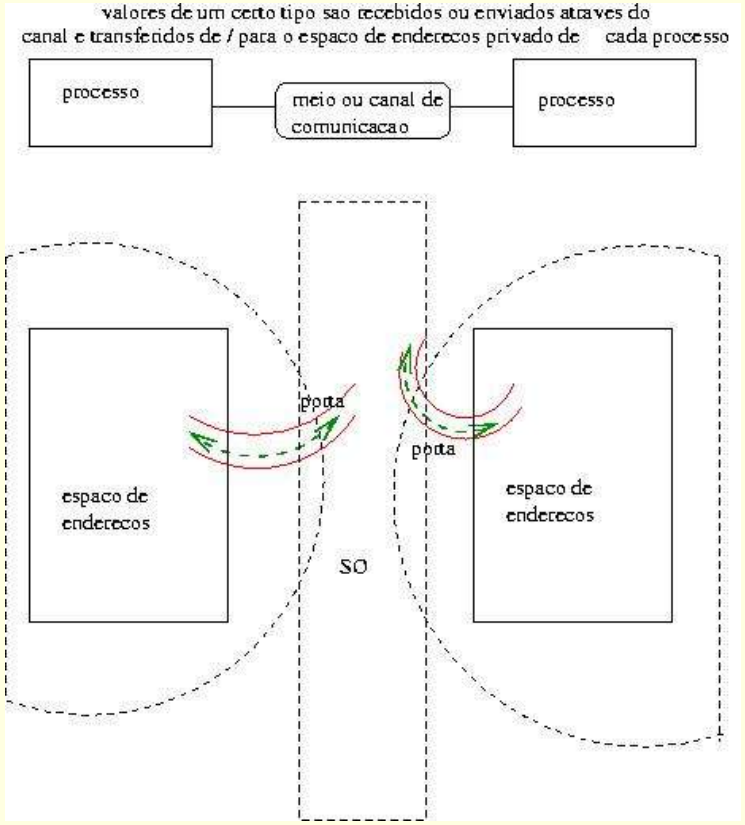
- *A API conhecida como IPC (Unix System V) para comunicação entre processos define, entre outros, mecanismos de comunicação como pipes e memória partilhada, bem como semáforos e vectores de semáforos.*
- *A norma POSIX (IEEE 1003.1c) define uma API para gestão de processos leves – criação, controle (prioridade, suspensão, terminar). Define também primitivas de sincronização: mutexes e semáforos generalizados, entre outras.*
- *Os problemas da programação concorrente são desafios complexos, e conceptualmente equivalentes quaisquer que sejam os mecanismos de controle de fluxo (processos vs. threads) e de comunicação (MP, semáforos, mutexes, pipes, etc.) usados.*

Comunicação entre processos

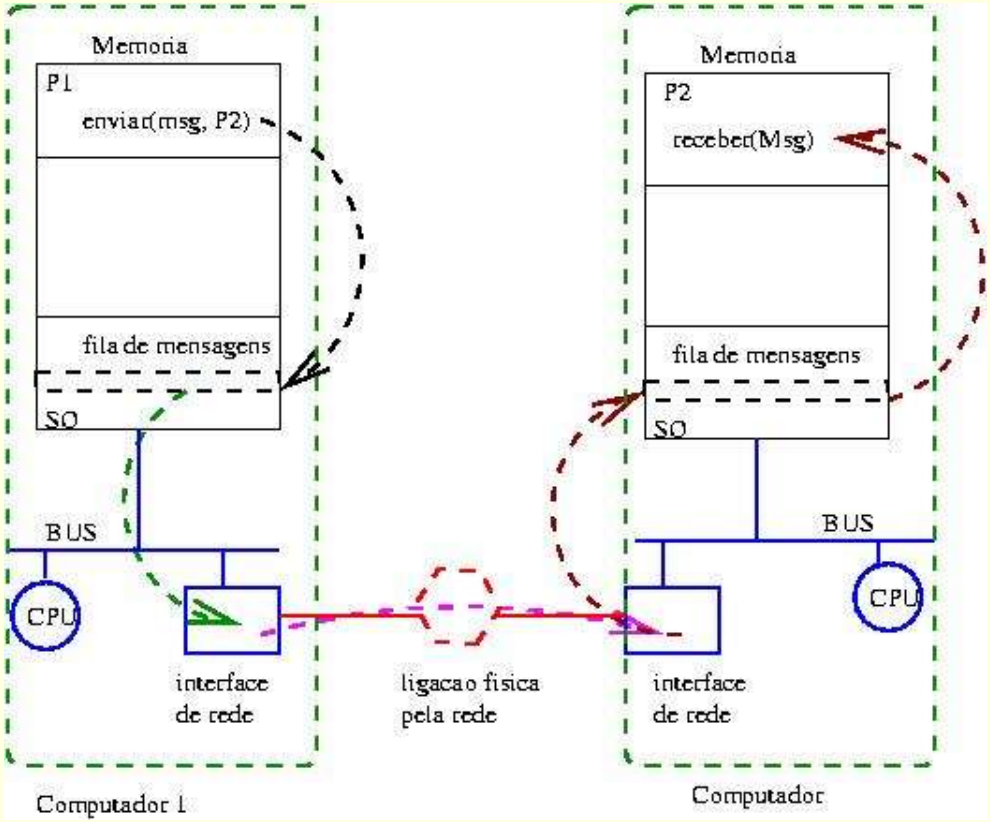
- Usando Memória Partilhada:
  - Acede-se aos dados lendo e escrevendo em memória,
  - Há uma só cópia dos dados,
  - É necessário sincronizar explicitamente os acessos,
  - Os processos envolvidos residem numa mesma máquina.
- Usando Memória Distribuída:
  - Enviam-se e recebem-se mensagens,
  - Com cópia de dados de um espaço de endereçamento para outro,
  - Com sincronização implícita,
  - Os processos envolvidos na comunicação podem residir em máquinas distintas (se interligadas por uma rede)

Comunicação por mensagens (1)





Comunicação por mensagens (2)

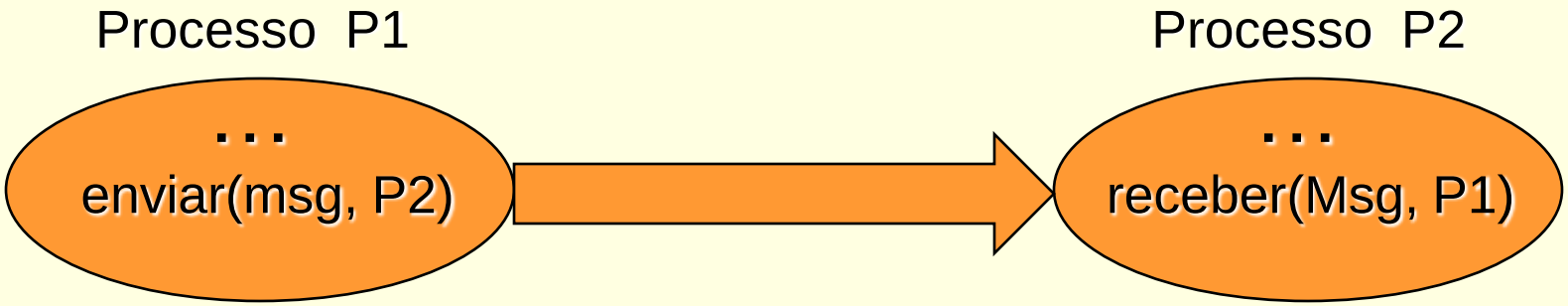


Dimensões da comunicação por mensagens

Unix Windows NT Netware MacOS DOS/VS Vax/VMS Linux Solaris HP/UX AIX Mach Chorus

1. Invocação: explícita vs. implícita
2. Nomeação: directa vs. indirecta
3. Padrão de interacção: 1:1, 1:N, N:1, N:M
4. Sincronia vs. Assincronia (operações bloqueantes ou não)
5. Suporte ao não-determinismo (selecção)
6. Fluxo de bytes (stream) vs. fluxo de mensagens
7. Papel dos participantes: simétrico vs. assimétrico (cliente/servidor)

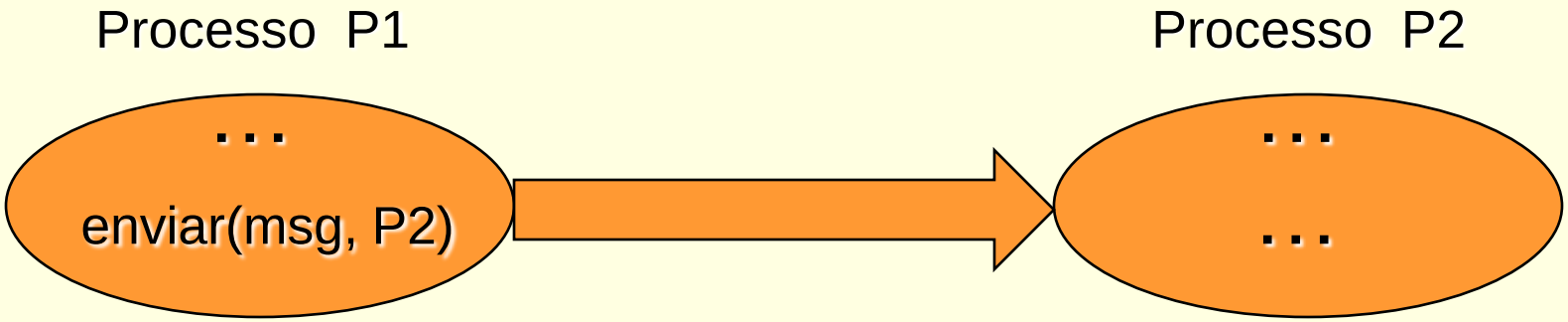
Invocação explícita



- Ambos têm de codificar, explicitamente, as primitivas de enviar e receber,
- Uma variante é o receptor aceitar mensagens de qualquer emissor, fazendo:

receber(Msg, qualquer)

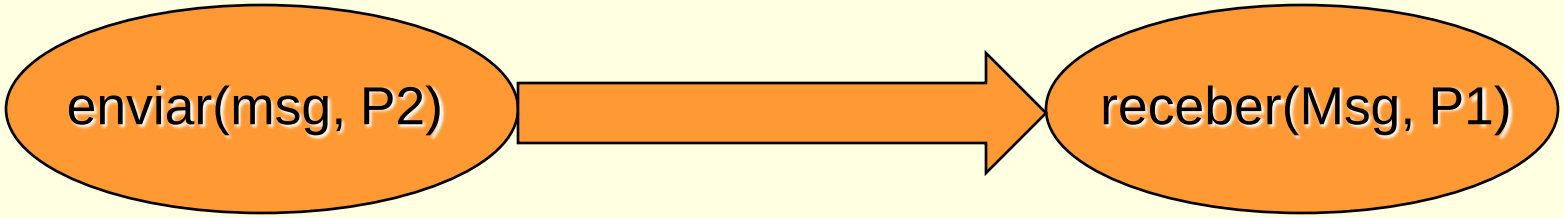
Invocação implícita



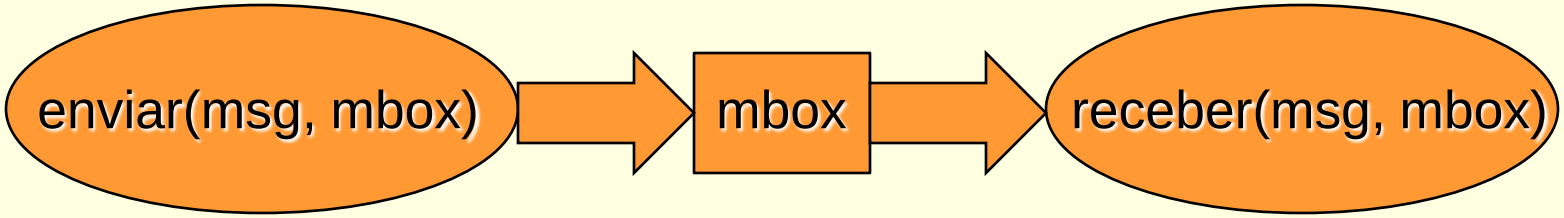
- O processo receptor define (tipicamente no início do programa) uma função - designada handler - que será invocada automaticamente quando chegar uma mensagem
- A execução da função é assíncrona, ou não-determinística relativamente ao código do programa: não se pode saber antecipadamente que parte do programa estará a ser executada no instante em que chega uma mensagem…

Nomeação directa vs. indirecta

- Nomeação directa: Codificação explícita dos identificadores dos intervenientes



- Nomeação indirecta: uso de um intermediário



Nomeação directa

- Na nomeação directa, os processos têm de se conhecer reciprocamente e têm de ser executados concorrentemente
  - Mesmo numa aplicação cliente/servidor este pressuposto acaba por se verificar, já que embora no início o servidor não conheça os seus clientes - e por isso recebe de qualquer um - depois da comunicação inicial tal já não se verifica.
  - O cliente, contudo, tem de nomear o servidor

Nomeação indirecta (1)

- Na nomeação indirecta:
  - Existe um intermediário na comunicação
  - Cada processo dirige-se ao intermediário para enviar e/ou receber mensagens
  - O intermediário tem, geralmente, um papel passivo:
    - Limita-se a gerir filas de mensagens, actuando como um distribuidor indirecto e como uma espécie de buffer activo;
    - aceita as mensagens que lhe enviam e distribui-as pelos destinatários, quando estes lhas pedirem.

Nomeação indirecta (2)

- Na nomeação indirecta, os processos não precisam de co-existir simultaneamente, podem ser eventualmente executados em momentos diferentes
- Facilita a interacção entre processos, de uma forma anónima:
  - qualquer processo pode enviar para a caixa e qualquer processo pode receber qualquer mensagem da caixa sem ter de ser explicitamente nomeado e conhecido pelo emissor
  - Por exemplo, pode ser usado para um qualquer processo no sistema notificar todos os outros ou um seu subconjunto, sem ter de os nomear e conhecer explicitamente
  - Qualquer novo processo que seja criado, pode começar a utilizar a caixa, desde que a esta tenha acesso

Exemplos de nomeação

Unix Windows NT Netware MacOS DOS/VS Vax/VMS Linux Solaris HP/UX AIX Mach Chorus

- São exemplos de comunicação por nomeação indirecta o uso de:
  - Pipes, já estudados
  - Message queues, a estudar na próxima aula
- São exemplos de comunicação por nomeação directa o uso de:
  - Sockets UDP e TCP