# Comunicação entre processos

António Pinto
apinto@estg.ipp.pt

Escola Superior de Tecnologia e Gestão

Novembro, 2017

# Sumário

# Conceitos introdutórios

## Processo cooperativo

Processo que partilha dados com outros processos e que, por isso, pode afetar ou ser afetado por estes.

# Conceitos introdutórios

## Processo cooperativo

Processo que partilha dados com outros processos e que, por isso, pode afetar ou ser afetado por estes.

- Forma mais elementar de partilha de dados ocorre por partilha de memória

# Conceitos introdutórios

## Processo cooperativo

Processo que partilha dados com outros processos e que, por isso, pode afetar ou ser afetado por estes.

- Forma mais elementar de partilha de dados ocorre por partilha de memória
- Partilha de memória só está disponível para processos que executem no mesmo PC

# Conceitos introdutórios

## Processo cooperativo

Processo que partilha dados com outros processos e que, por isso, pode afetar ou ser afetado por estes.

- Forma mais elementar de partilha de dados ocorre por partilha de memória
- Partilha de memória só está disponível para processos que executem no mesmo PC
- Não é possível entre processos que não usem o mesmo espaço de endereçamento de memória

# Conceitos introdutórios

## Processo cooperativo

Processo que partilha dados com outros processos e que, por isso, pode afetar ou ser afetado por estes.

- Forma mais elementar de partilha de dados ocorre por partilha de memória
- Partilha de memória só está disponível para processos que executem no mesmo PC
- Não é possível entre processos que não usem o mesmo espaço de endereçamento de memória
- É necessário mecanismo alternativo de comunicação entre processos

# Formas de comunicação

- Uma forma de comunicação entre processos que não partilhem memória pode ser conseguida com um mecanismo de troca de mensagens

# Formas de comunicação

- Uma forma de comunicação entre processos que não partilhem memória pode ser conseguida com um mecanismo de troca de mensagens
- Troca de mensagens pode ocorrer também entre processos a correr em equipamentos distintos

# Formas de comunicação

- Uma forma de comunicação entre processos que não partilhem memória pode ser conseguida com um mecanismo de troca de mensagens
- Troca de mensagens pode ocorrer também entre processos a correr em equipamentos distintos
- Internet, com os pacotes IP, é exemplo de um sistema de troca de mensagens global

# Conteúdos

# Sistema de troca de mensagens

- Funcionalidade elementar deste sistema é o de permitir comunicação entre processos, sem recorrer à partilha de memória

# Sistema de troca de mensagens

- Funcionalidade elementar deste sistema é o de permitir comunicação entre processos, sem recorrer à partilha de memória
- Deve providenciar, pelo menos, as seguintes funções
  - *enviar(mensagem)*
  - *receber(mensagem)*

# Sistema de troca de mensagens

- Funcionalidade elementar deste sistema é o de permitir comunicação entre processos, sem recorrer à partilha de memória
- Deve providenciar, pelo menos, as seguintes funções
  - *enviar(mensagem)*
  - *receber(mensagem)*
- Mensagens podem ter tamanho fixo ou variável
  - Fixo é mais fácil de implementar pelo sistema operativo
  - Variável é mais útil para quem desenvolve aplicações

# Sistema de troca de mensagens

## Ligação

- Comunicação entre dois quaisquer processos requer que consigam mandar e receber mensagens entre eles

# Sistema de troca de mensagens

## Ligação

- Comunicação entre dois quaisquer processos requer que consigam mandar e receber mensagens entre eles
- Uma **ligação** entre estes é necessária

# Sistema de troca de mensagens

## Ligação

- Comunicação entre dois quaisquer processos requer que consigam mandar e receber mensagens entre eles
- Uma **ligação** entre estes é necessária
- Ligação pode ser implementada de muitas formas

# Sistema de troca de mensagens

## Ligação

- Comunicação entre dois quaisquer processos requer que consigam mandar e receber mensagens entre eles
- Uma **ligação** entre estes é necessária
- Ligação pode ser implementada de muitas formas
- Existem fatores que afetam a forma como uma ligação é utilizada/implementada
  - Se a comunicação é **direta ou indireta**
  - Se a comunicação é **síncrona ou assíncrona**
  - Se a comunicação usa ***buffering* ou não**

# Conteúdos

# Endereçamento

- Processos para comunicar, necessitam de uma forma de se identificarem (*endereços*)

# Endereçamento

- Processos para comunicar, necessitam de uma forma de se identificarem (*endereços*)
- Uso de endereçamento tem impacto nos tipos de comunicação disponíveis
  - Comunicação direta
  - Comunicação indireta

# Comunicação direta

## Endereçamento simétrico

- Em **comunicação direta**, cada processo comunicante têm de explicitamente de identificar o outro processo

# Comunicação direta

## Endereçamento simétrico

- Em **comunicação direta**, cada processo comunicante têm de explicitamente de identificar o outro processo
  - *enviar(P,M)* $\rightarrow$ Enviar mensagem M para processo P
  - *receber(Q,M)* $\rightarrow$ Receber mensagem M do processo Q
  - Denominado de endereçamento **simétrico**

# Comunicação direta

## Endereçamento simétrico

- Em **comunicação direta**, cada processo comunicante têm de explicitamente de identificar o outro processo
  - *enviar(P,M)* $\rightarrow$ Enviar mensagem M para processo P
  - *receber(Q,M)* $\rightarrow$ Receber mensagem M do processo Q
  - Denominado de endereçamento **simétrico**
- Características deste tipo de ligação
  - Processo necessitam conhecer endereços uns dos outros
  - Ligação associada a exatamente 2 processos
  - Só permite uma ligação entre 2 processos

# Comunicação direta

## Endereçamento assimétrico

- Quando é permitido ao processo receber mensagens sem conhecer previamente o emissor, estamos perante endereçamento **assimétrico**

# Comunicação direta

## Endereçamento assimétrico

- Quando é permitido ao processo receber mensagens sem conhecer previamente o emissor, estamos perante endereçamento **assimétrico**
  - *enviar(P,M)* $\rightarrow$ Enviar mensagem M para processo P
  - *receber(?,M)* $\rightarrow$ Receber mensagem M de qualquer processo. Mensagem inclui identificação processo emissor

# Comunicação indireta

- Comunicação pode ser efetuada com recurso a armazenamento temporário

# Comunicação indireta

- Comunicação pode ser efetuada com recurso a armazenamento temporário
- Mensagens são armazenadas num **repositório** fora do processo

# Comunicação indireta

- Comunicação pode ser efetuada com recurso a armazenamento temporário
- Mensagens são armazenadas num **repositório** fora do processo
- Repositórios passam a ter endereços

# Comunicação indireta

- Comunicação pode ser efetuada com recurso a armazenamento temporário
- Mensagens são armazenadas num **repositório** fora do processo
- Repositórios passam a ter endereços
- Dois processos comunicam partilhando um repositório
  - *enviar(R,M)* $\rightarrow$ Enviar mensagem M para repositório R
  - *receber(R,M)* $\rightarrow$ Receber mensagem M de repositório R

# Comunicação indireta

## Características

- Ligação só é estabelecida entre 2 processos se estes partilharem um repositório

# Comunicação indireta

## Características

- Ligação só é estabelecida entre 2 processos se estes partilharem um repositório
- Ligação pode servir mais do que 2 processos

# Comunicação indireta

## Características

- Ligação só é estabelecida entre 2 processos se estes partilharem um repositório
- Ligação pode servir mais do que 2 processos
- Podem existir múltiplas ligações (repositórios) entre cada 2 processos

# Comunicação indireta

Utilização de repositório por mais do que 2 processos

Assumindo que

- Processos $P_1$,$P_2$ e $P_3$ partilham o repositório A
- $P_1$ envia mensagem para o repositório A
- $P_2$ e $P_3$ executam *receber(A,M)*

Quem recebe mensagem?

# Comunicação indireta

Utilização de repositório por mais do que 2 processos

Assumindo que

- Processos $P_1$,$P_2$ e $P_3$ partilham o repositório A
- $P_1$ envia mensagem para o repositório A
- $P_2$ e $P_3$ executam *receber(A,M)*

Quem recebe mensagem? $\rightarrow$ **Depende**

# Comunicação indireta

Utilização de repositório por mais do que 2 processos

Assumindo que

- Processos $P_1$,$P_2$ e $P_3$ partilham o repositório A
- $P_1$ envia mensagem para o repositório A
- $P_2$ e $P_3$ executam *receber(A,M)*

Quem recebe mensagem? $\rightarrow$ **Depende**

- Pode-se limitar partilha de repositório a 2 processos

# Comunicação indireta

Utilização de repositório por mais do que 2 processos

Assumindo que

- Processos $P_1$,$P_2$ e $P_3$ partilham o repositório A
- $P_1$ envia mensagem para o repositório A
- $P_2$ e $P_3$ executam *receber(A,M)*

Quem recebe mensagem? $\rightarrow$ **Depende**

- Pode-se limitar partilha de repositório a 2 processos
- Pode-se impedir a execução simultânea de *receber(A,M)*

# Comunicação indireta

Utilização de repositório por mais do que 2 processos

Assumindo que

- Processos $P_1$,$P_2$ e $P_3$ partilham o repositório A
- $P_1$ envia mensagem para o repositório A
- $P_2$ e $P_3$ executam *receber(A,M)*

Quem recebe mensagem? $\rightarrow$ **Depende**

- Pode-se limitar partilha de repositório a 2 processos
- Pode-se impedir a execução simultânea de *receber(A,M)*
- Pode-se deixar o sistema escolher aleatoriamente

# Comunicação indireta

## Autoridade sobre repositórios

- Repositórios podem pertencer ao sistema operativo ou ao processo

# Comunicação indireta

## Autoridade sobre repositórios

- Repositórios podem pertencer ao sistema operativo ou ao processo
- Pertença ao processo
  - Apenas o processo pode ler do repositório
  - Repositório é eliminado aquando do término do processo

# Comunicação indireta

## Autoridade sobre repositórios

- Repositórios podem pertencer ao sistema operativo ou ao processo
- Pertença ao processo
  - Apenas o processo pode ler do repositório
  - Repositório é eliminado aquando do término do processo
- Pertença ao sistema operativo
  - Repositório tem existência própria
  - Requer funções de gestão de repositórios (criar, eliminar, editar, . . . )

# Conteúdos

# Sincronismo

- Comunicação entre processos ocorre com a utilização das funções *enviar()* e *receber()*

# Sincronismo

- Comunicação entre processos ocorre com a utilização das funções *enviar()* e *receber()*
- Estas funções podem ser implementadas adotando-se um abordagem **bloqueante** (síncrona) ou **não bloqueante** (assíncrona)

# Sincronismo

## Funções

- **enviar() bloqueante**: Função de envio só termina quando a mensagem é recebida pelo destinatário (processo ou repositório)

# Sincronismo

## Funções

- **enviar() bloqueante**: Função de envio só termina quando a mensagem é recebida pelo destinatário (processo ou repositório)
- **enviar() não bloqueante**: Função de envio termina mal acaba de enviar a mensagem

# Sincronismo

## Funções

- **enviar() bloqueante**: Função de envio só termina quando a mensagem é recebida pelo destinatário (processo ou repositório)
- **enviar() não bloqueante**: Função de envio termina mal acaba de enviar a mensagem
- **receber() bloqueante**: Função de receção só termina quando for efetivamente recebida uma mensagem pelo destinatário

# Sincronismo

## Funções

- **enviar() bloqueante**: Função de envio só termina quando a mensagem é recebida pelo destinatário (processo ou repositório)
- **enviar() não bloqueante**: Função de envio termina mal acaba de enviar a mensagem
- **receber() bloqueante**: Função de receção só termina quando for efetivamente recebida uma mensagem pelo destinatário
- **receber() não bloqueante**: Função de receção verifica se existe mensagem, terminando de imediato (devolvendo a mensagem ou *null*)

# Sincronismo

## Funções

- Qualquer combinação destas funções é possível

# Sincronismo

## Funções

- Qualquer combinação destas funções é possível
- Quando *enviar()* e *receber()* são bloqueantes
  - Dá-se o ***rendezvous*** dos processos
  - Ambos os processo têm de estar a executar em simultâneo para que se dê a comunicação

# *Conteúdos*

# Buffering

- Capacidade de comunicação está associada a utilização de zonas de armazenamento de informação

# Buffering

- Capacidade de comunicação está associada a utilização de zonas de armazenamento de informação
- O armazenamento é temporário já que acontece só até ao momento de leitura

# Buffering

- Capacidade de comunicação está associada a utilização de zonas de armazenamento de informação
- O armazenamento é temporário já que acontece só até ao momento de leitura
- Zonas de armazenamento temporário são organizadas como filas (*queues*)

# Buffering

- Capacidade de comunicação está associada a utilização de zonas de armazenamento de informação
- O armazenamento é temporário já que acontece só até ao momento de leitura
- Zonas de armazenamento temporário são organizadas como filas (*queues*)
- Existem 3 tipos de filas
  - Filas com capacidade zero
  - Filas com capacidade limitada
  - Filas com capacidade ilimitada

# Buffering

## Filas com capacidade zero

- Fila tem 0 (zero) capacidade de armazenamento
- Não é capaz de conter mensagens

# Buffering

## Filas com capacidade zero

- Fila tem 0 (zero) capacidade de armazenamento
- Não é capaz de conter mensagens
- Emissor bloqueia até que a mensagem seja lida
- Implica o *rendezvous*

# Buffering

## Filas com capacidade limitada

- Fila tem capacidade de armazenamento de algumas mensagens

# Buffering

## Filas com capacidade limitada

- Fila tem capacidade de armazenamento de algumas mensagens
- Fila tem limite máximo de mensagens

# Buffering

## Filas com capacidade limitada

- Fila tem capacidade de armazenamento de algumas mensagens
- Fila tem limite máximo de mensagens
- Se fila não estiver cheia, a mensagem é armazenada e o emissor pode continuar
- Se fila estiver cheia, emissor tem de aguardar até que seja libertado espaço para armazenar a mensagem

# Buffering

## Filas com capacidade ilimitada

- Fila tem capacidade de armazenamento infinito
- Emissor nunca bloqueia na operação de envio de mensagem

# Bibliografia

- Baseado na bibliografia da unidade curricular.