

BCC264

Sistemas Operacionais

Processos

Prof. Charles Garrocho

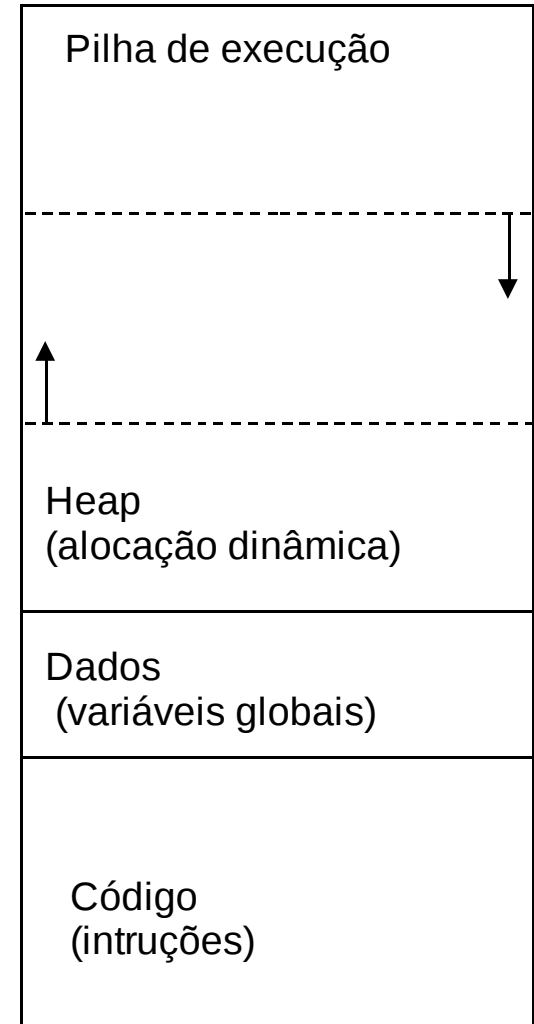
O conceito de processo

- Um S.O. executa uma variedade de programas
 - Sistemas de tempo compartilhado: programas
- Processo: um programa em execução
 - Um fluxo de controle de execução de instruções
 - Um contador de programa
 - Uma pilha de chamadas
 - Uma área de dados (memória)

Etapas de um processo

- Criação
 - Inicialização
 - Definição da imagem na memória:
- Execução
 - Escalonamento pelo núcleo
 - Comunicação entre processos
 - Acesso a dispositivos
- Terminação
 - Liberação de recursos

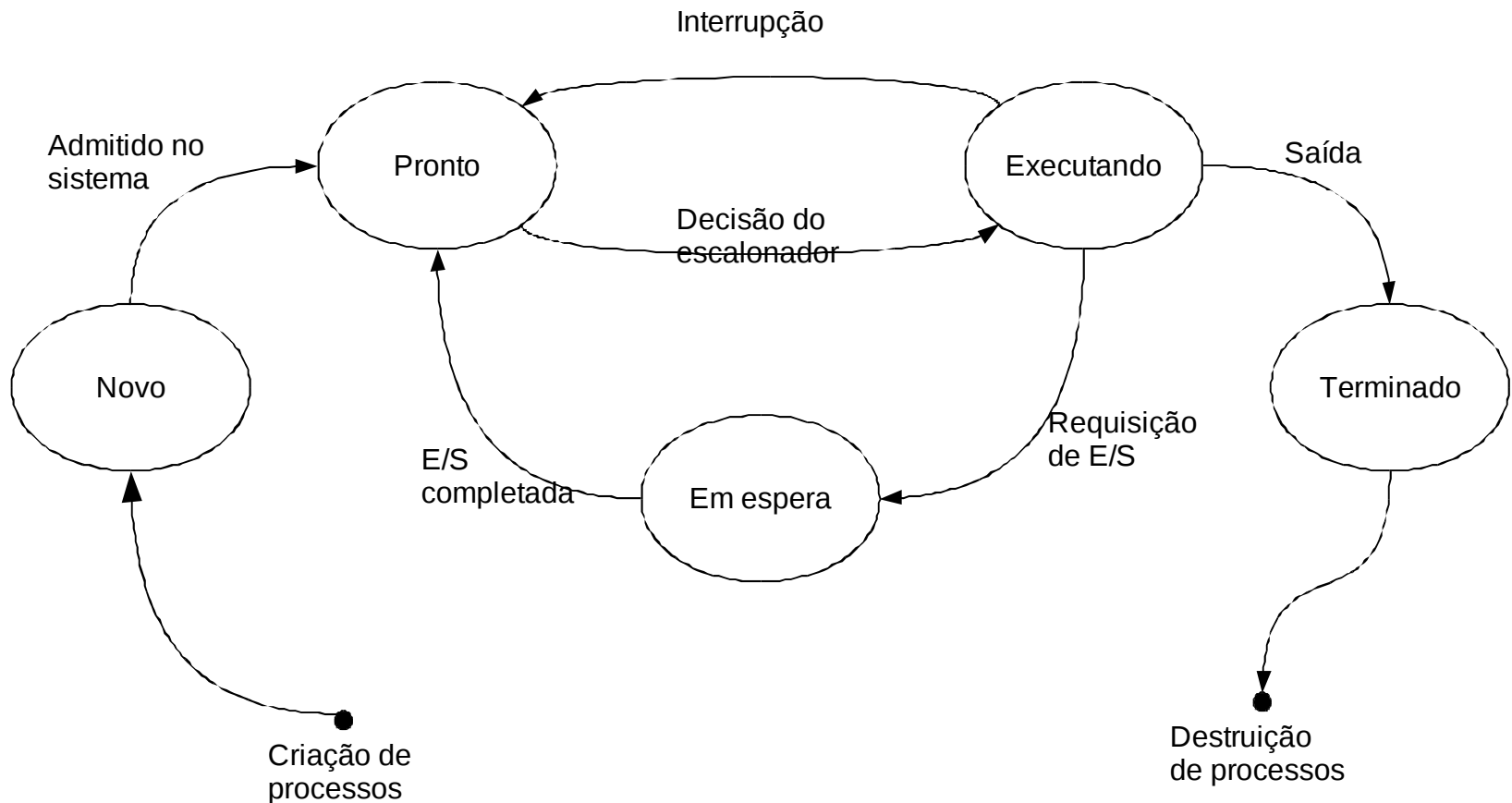
Mapa de memória



Estado de um processo

- Condição do processo no S.O.:
 - novo: o processo acaba de ser criado
 - executando: CPU está executando suas instruções
 - em espera: processo aguarda algum evento
 - pronto: o processo espera pela CPU
 - terminado: processo completou sua execução

Diagrama de estados de processo



Bloco de controle de processo (PCB)

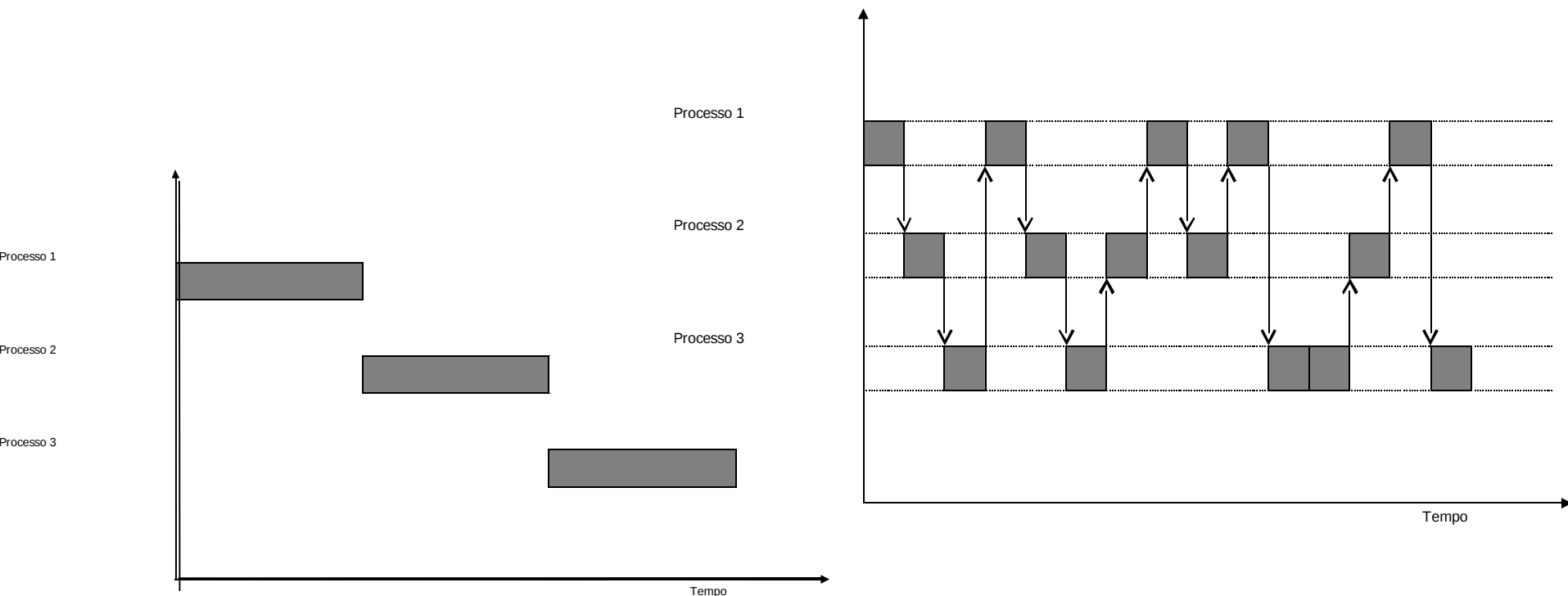
- Informação associada com cada processo
 - Estado do processo
 - Contador de programa associado
 - Conteúdo dos registradores da CPU
 - Informações de escalonamento da CPU
 - Informações de gerência de memória
 - Informações de contabilidade
 - Informações sobre estado de eventos de E/S

Bloco de controle de processo (PCB)

pointer	process state
process number	
program counter	
registers	
memory limits	
list of open files	
• • •	

Escalonamento de processos

- Aumentar a utilização do processador
- Paralelismo aparente e transparente
- O S.O. decide quem, quando e como!



Filas de escalonamento de processos

- Fila de jobs – conjunto de todos os processos
- Fila de prontos – todos os processos presentes na memória, prontos e esperando pela CPU
 - Podem haver várias filas de prontos com níveis de prioridades diferentes
- Filas de dispositivos – processos aguardando pela resposta de um dispositivo de E/S

Escalonadores

- Escalonadores de longo prazo (de jobs)
 - Seleciona os processos a serem inseridos na fila de prontos (entrar no sistema)
- Escalonadores de curto prazo (de CPU)
 - Seleciona o processo a receber a CPU a cada instante

Escalonadores

- Escalonador de CPU é chamado frequentemente (mili-segundos)
 - deve ser muito rápido
- Escalonador de jobs é chamado com menor frequência (segundos, minutos)
 - pode ser mais lento e usar mais informação
 - controla o grau de multiprogramação (no. de processos que podem executar “em paralelo”)

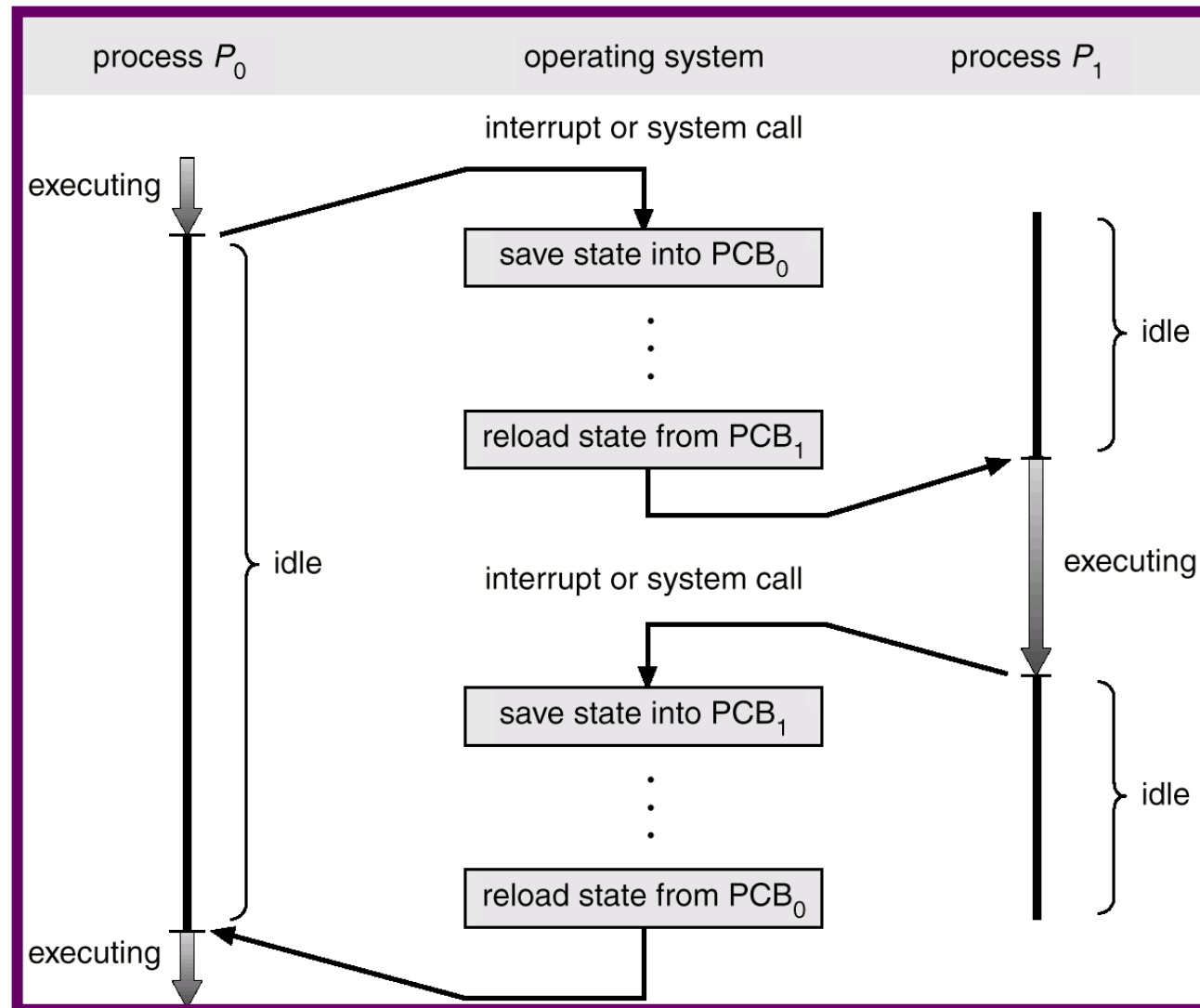
Escalonadores

- Processo podem ser descritos como:
 - “I/O bound” (intensivos em E/S)
 - passa mais tempo esperando por E/S que computando
 - muitas rajadas curtas de processamento
 - “CPU bound” (intensivos em processamento)
 - passa mais tempo em processamento
 - períodos longos de processamento, na CPU

Troca/chaveamento de contexto (*context switch*)

- Mudança do processo em execução
- O S.O. deve salvar o estado do processo atual e carregar o estado do novo processo
- Troca de contexto é *overhead*
- Tempo gasto depende do hardware e da estrutura do processo no S.O.

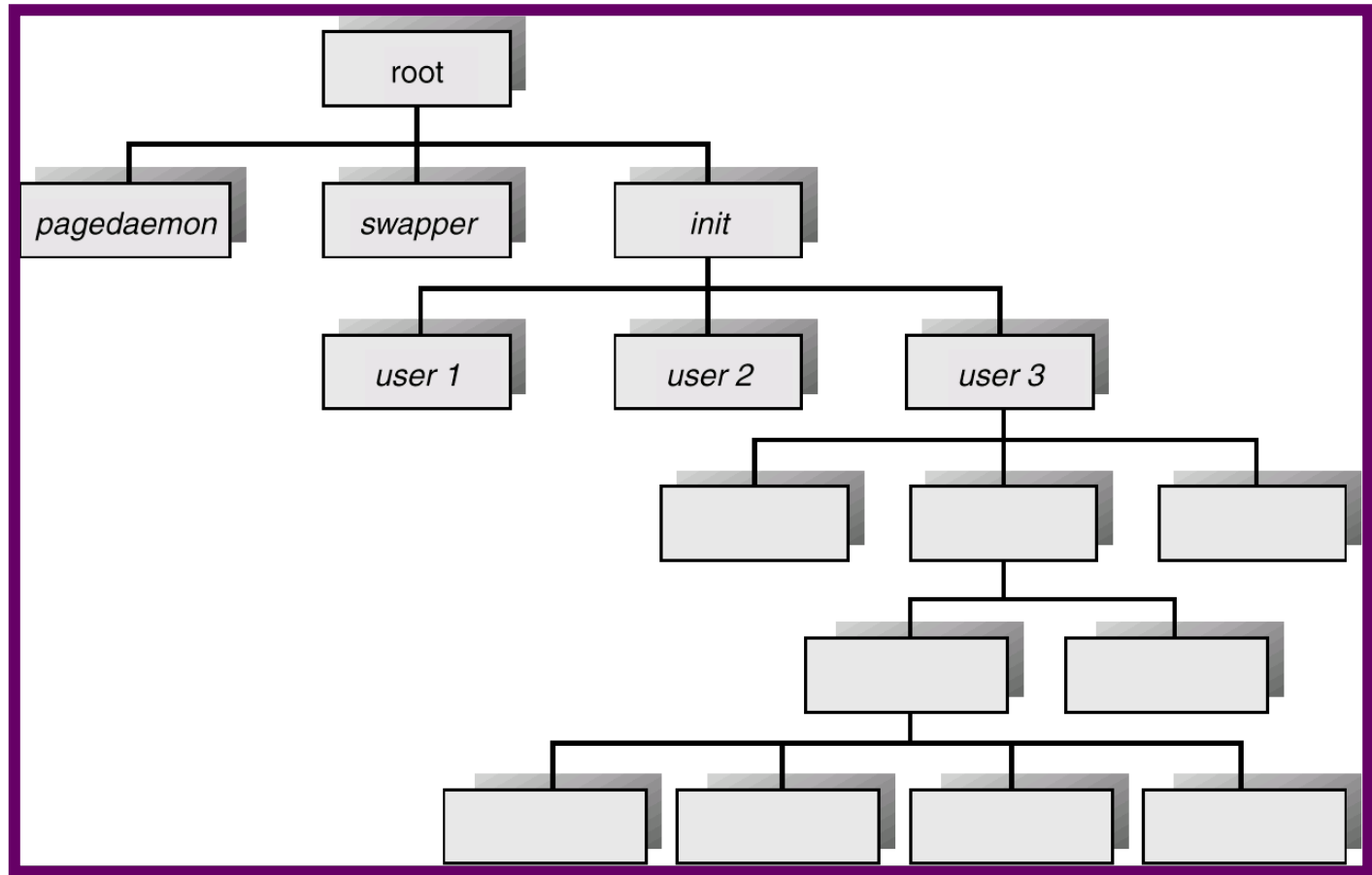
Troca/chaveamento de contexto (*context switch*)



Criação de processos

- Um processo cria (pai) outros (filhos), que por sua vez podem criar ainda outros
 - Estabelece-se uma árvore de processos
 - Filhos podem ser cópias do processo pai ou derivados de outros programas
 - Diversos níveis de compartilhamento são possíveis entre níveis da árvores (pais e filhos)
 - Execução de pais e filhos pode ser concorrente ou pais podem esperar pelos filhos

Árvore de processos no Unix



Terminação de processos

- Ao terminar seu processamento, processo pede ao S.O. para retirá-lo da fila de jobs
 - O pedido pode ser bem comportado ou causado por erros de execução
 - O processo criador (pai) é informado de seu fim
 - Os recursos do processo são liberados pelo S.O.

Terminação de processos

- O S.O. ou o processo pai pode interromper a execução de processos filhos
 - O filho pode ter excedido os recursos alocados
 - A tarefa do filho pode não ser mais necessária
 - O pai pode ter que terminar ele mesmo
 - sem um comando especial, o S.O. não permite que processos filhos continuem sem seus pais
 - terminação em cascata

Cooperação entre processos

- Processos independentes não podem afetar ou ser afetados pela execução de outros
- Vantagens da cooperação entre processos
 - Compartilhamento de informação
 - Aumento da velocidade de processamento
 - Modularidade
 - Conveniência

Problema do produtor/consumidor

- Modelo básico de cooperação geral
 - Produtor produz informação e entrega ao cons.
 - Consumidor utiliza informação recebida
- Comunicação pode ser feita por um buffer
 - Buffer de tamanho limitado impõe restrições
 - Mesmo buffer ilimitado deve caber na memória
- Comunicação também pode ser por mensagens

Comunicação entre processos (IPC)

- Sistema de troca de mensagens, sem memória compartilhada
 - comunicação e sincronização
- Baseado em duas operações básicas:
 - `send(message)` – tamanho fixo ou variável
 - `receive(message)`

Comunicação entre processos (IPC)

- Se P e Q querem se comunicar, eles devem:
 - estabelecer um canal de comunicação
 - trocar mensagens com send/receive
- Implementação do canal de comunicação
 - escolha do meio: barramento de HW, memória
 - comportamento lógico: como funciona

Comunicação direta

- Processos identificam o outro explicitamente
 - `send (P, message)` – envia msg p/ processo P
 - `receive(Q, message)` – recebe msg do proc. Q
 - recepção pode usar máscara, como “Q=qualquer um”

Comunicação direta

- Propriedades do canal de comunicação direta
 - Estabelecimento é automático
 - Cada canal liga exatamente um par de processos
 - Existe apenas um canal entre cada par
 - Canais são usualmente bi-direcionais

Comunicação indireta

- Mensagens são originadas de e direcionadas para caixas de correio (*ports* = portas/portos)
 - Cada caixa tem um identificador único
 - Processos só se comunicam se compartilharem uma caixa de correio
 - Caixas de correio são recursos independentes que precisam ser criados e destruídos

Comunicação indireta

- Propriedades dos canais
 - Um canal pode ligar vários processos
 - Cada par de processos pode ter vários canais
 - Canais podem ser uni- ou bi-direcionais

Comunicação indireta

- Compartilhamento de caixas de correio
 - P1, P2 e P3 compartilham a caixa de correio A
 - P1 envia uma mensagem; quem recebe?
- Diversas soluções são possíveis
 - Restringir o compartilhamento a dois processos
 - Permitir que só um processo chame receive de cada vez
 - Escolher arbitrariamente quem recebe a msg
 - Entregar uma cópia a cada receive

Sincronização de primitivas

- Troca de mensagens: bloqueante ou não
 - Bloqueante: síncrona
 - Não bloqueante: assíncrona
- Cada primitiva (send/receive) pode se comportar de uma forma. Mais comum:
 - send assíncrono (retorna imediatamente)
 - receive síncrono (bloqueia até mensagem chegar)