DCC062 – Sistemas Operacionais

Cap. 3 – Deadlocks

Parte 1

Prof. Eduardo Pagani Julio

eduardo.pagani@ice.ufjf.br

Compartilhamento de Recursos

- Exemplos de recursos de computador
  - Impressoras
  - Unidades de fita
  - Tabelas
- Processos precisam de acesso aos recursos numa ordem racional
  - Suponha que um processo detenha o recurso A e solicite o recurso B
  - Ao mesmo tempo um outro processo detém B e solicita A
  - Ambos são bloqueados e assim permanecem

Recursos

- Temos que nos preocupar com o compartilhamento de recursos
  - Recursos podem ser dispositivos ou qualquer item compartilhado
  - Deadlocks podem ocorrer quando garante-se aos processos acesso exclusivo a recursos
- Recursos preemptíveis
  - Podem ser retirados de um processo sem quaisquer efeitos prejudiciais
- Recursos não preemptíveis
  - Vão induzir o processo a falhar se forem retirados

Recursos

- Seqüência de eventos necessários ao uso de um recurso
  - solicitar o recurso
  - usar o recurso
  - liberar o recurso
- Deve esperar se solicitação é negada
  - processo solicitante pode ser bloqueado
  - pode falhar resultando em um código de erro

Conceito de Deadlock

- Definição formal:
  - Um conjunto de processos está em situação de impasse se todo processo pertencente ao conjunto estiver esperando por um evento que somente um outro processo desse mesmo conjunto poderá fazer acontecer
- Normalmente o evento é a liberação de um recurso atualmente retido
- Nenhum dos processos consegue...
  - executar
  - liberar recursos
  - ser acordado

Condições para Deadlock

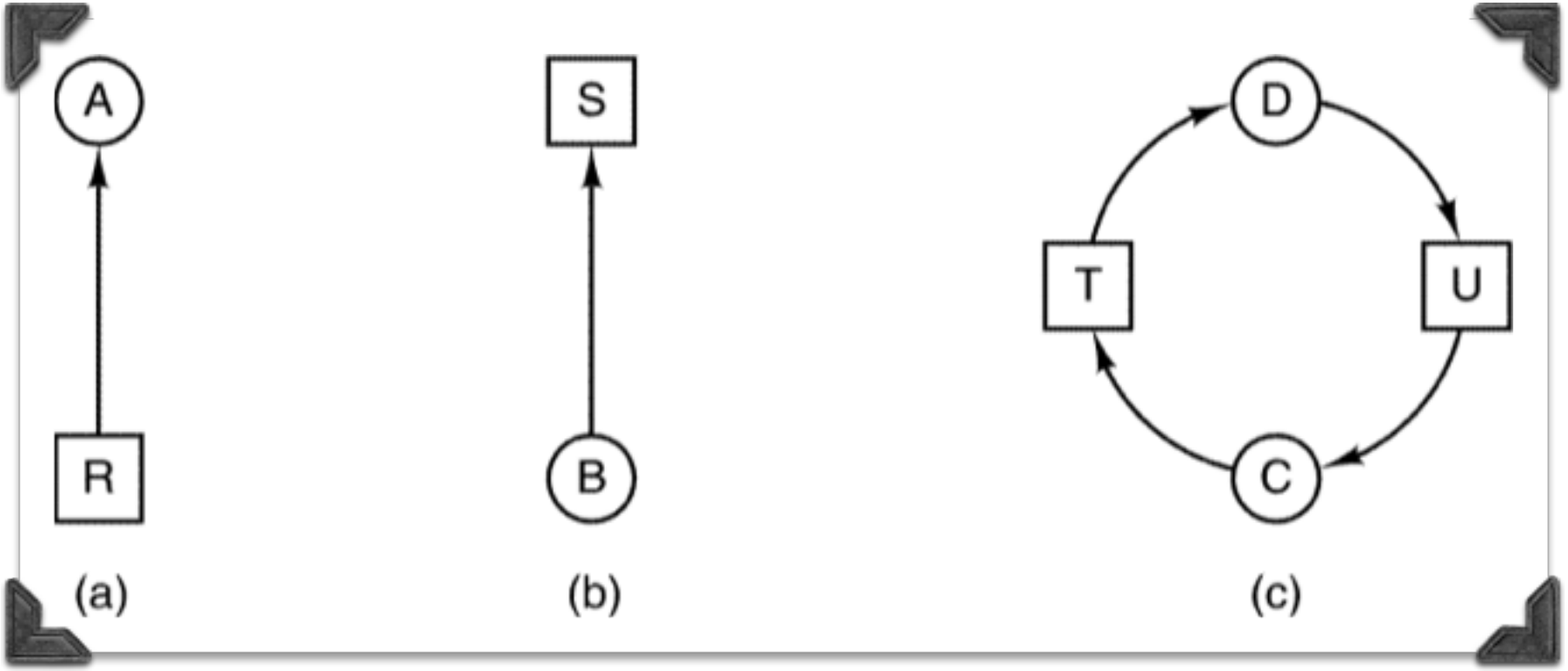
- Condição de exclusão mútua
  - todo recurso está ou associado a um único processo ou disponível
- Condição de posse e espera
  - processos que retêm recursos podem solicitar novos recursos
- Condição de não preempção
  - recursos concedidos previamente não podem ser forçosamente tomados
- Condição de espera circular
  - deve ser uma cadeia circular de 2 ou mais processos
  - cada um está à espera de recurso retido pelo membro seguinte dessa cadeia

O que fazer?

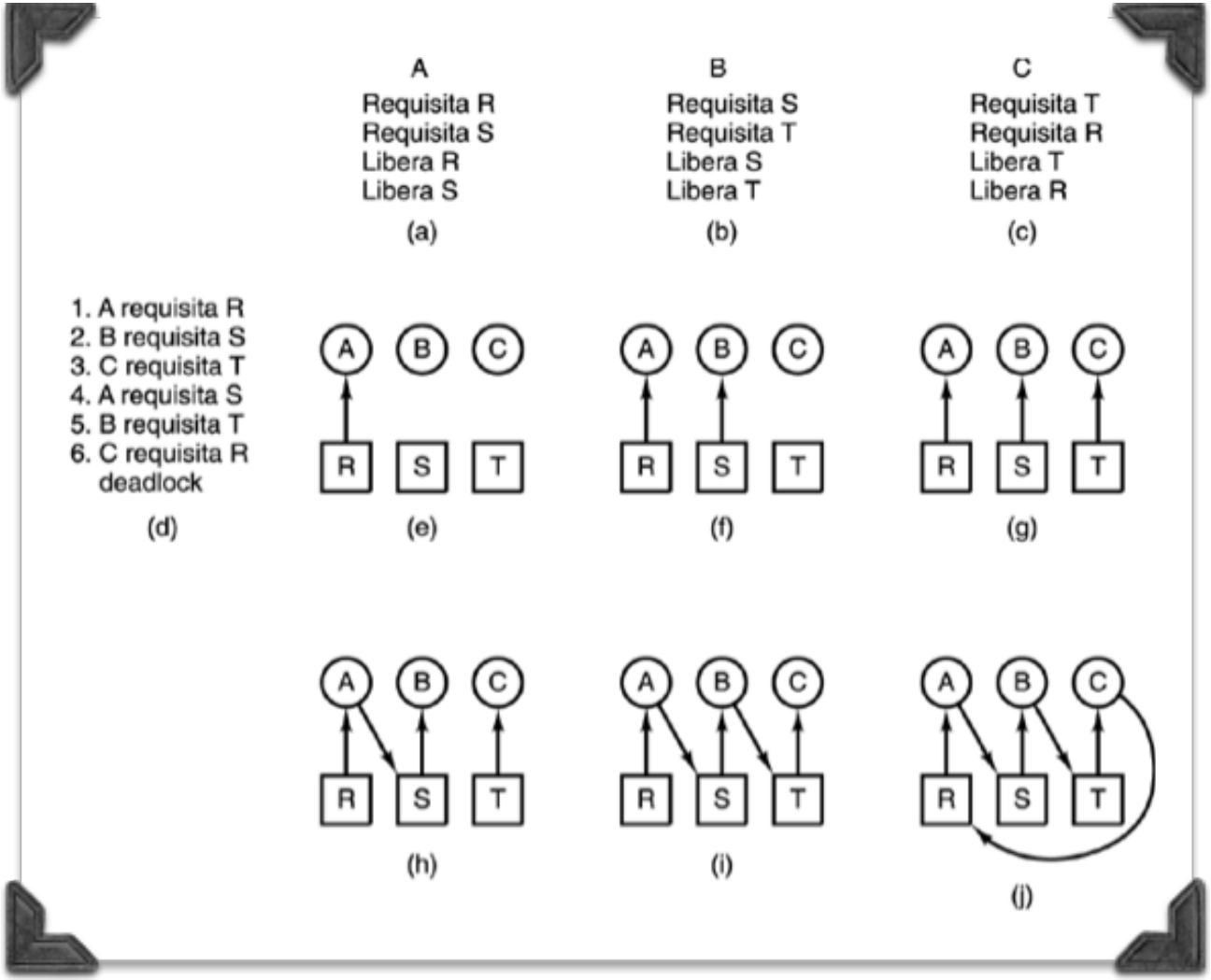
- Estratégias para tratar Deadlocks
  - ignorar por completo o problema
  - detecção e recuperação
  - evitar dinamicamente
    - alocação cuidadosa de recursos
  - prevenção
    - negação de uma das quatro condições necessárias

Modelagem de Deadlock

- Grafo dirigido



Modelagem de Deadlock - Ocorrência

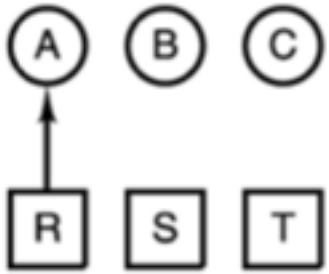


Modelagem de Deadlock - Não-ocorrencia

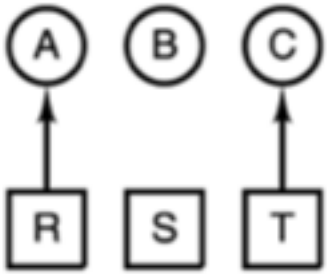
1. A requisita R
2. C requisita T
3. A requisita S
4. C requisita R
5. A libera R
6. A libera S

nenhum deadlock

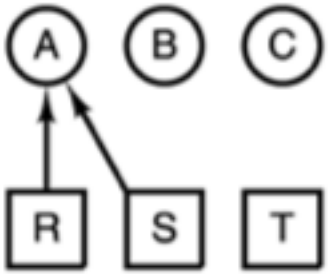
(k)



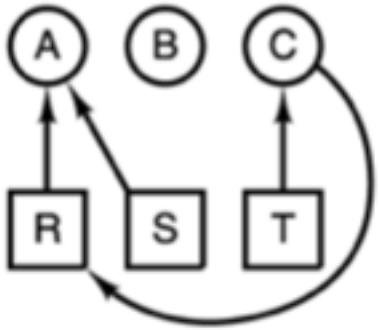
(l)



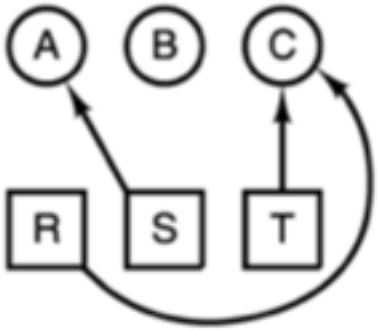
(m)



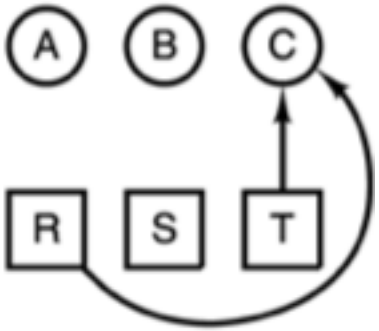
(n)



(o)



(p)

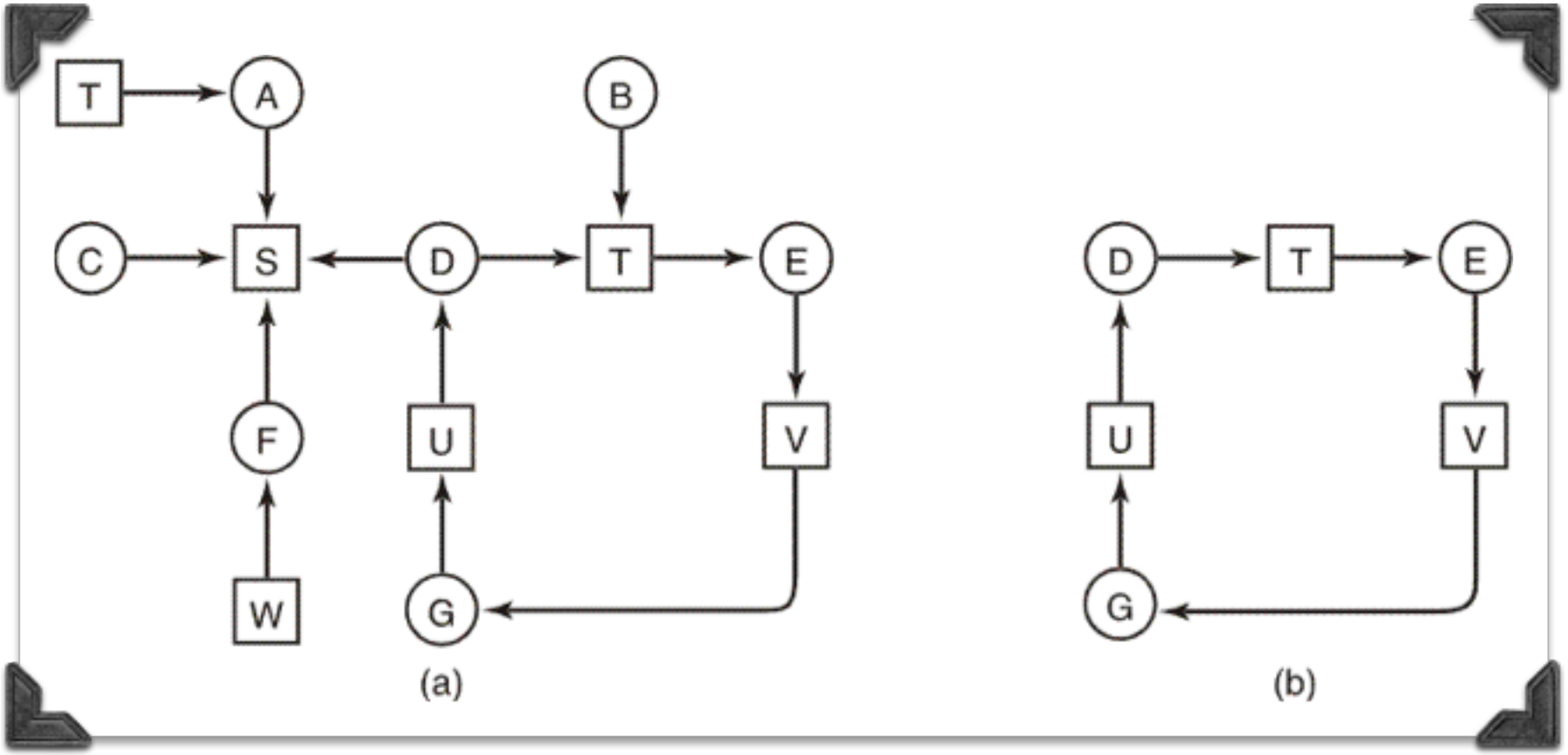


(q)

Estratégia prática

- Algoritmo do Avestruz
  - Finge que o problema não existe
  - Razoável se deadlocks ocorrem muito raramente
    - custo da prevenção é alto
  - UNIX e Windows seguem esta abordagem
  - É uma ponderação entre
    - conveniência
    - correção

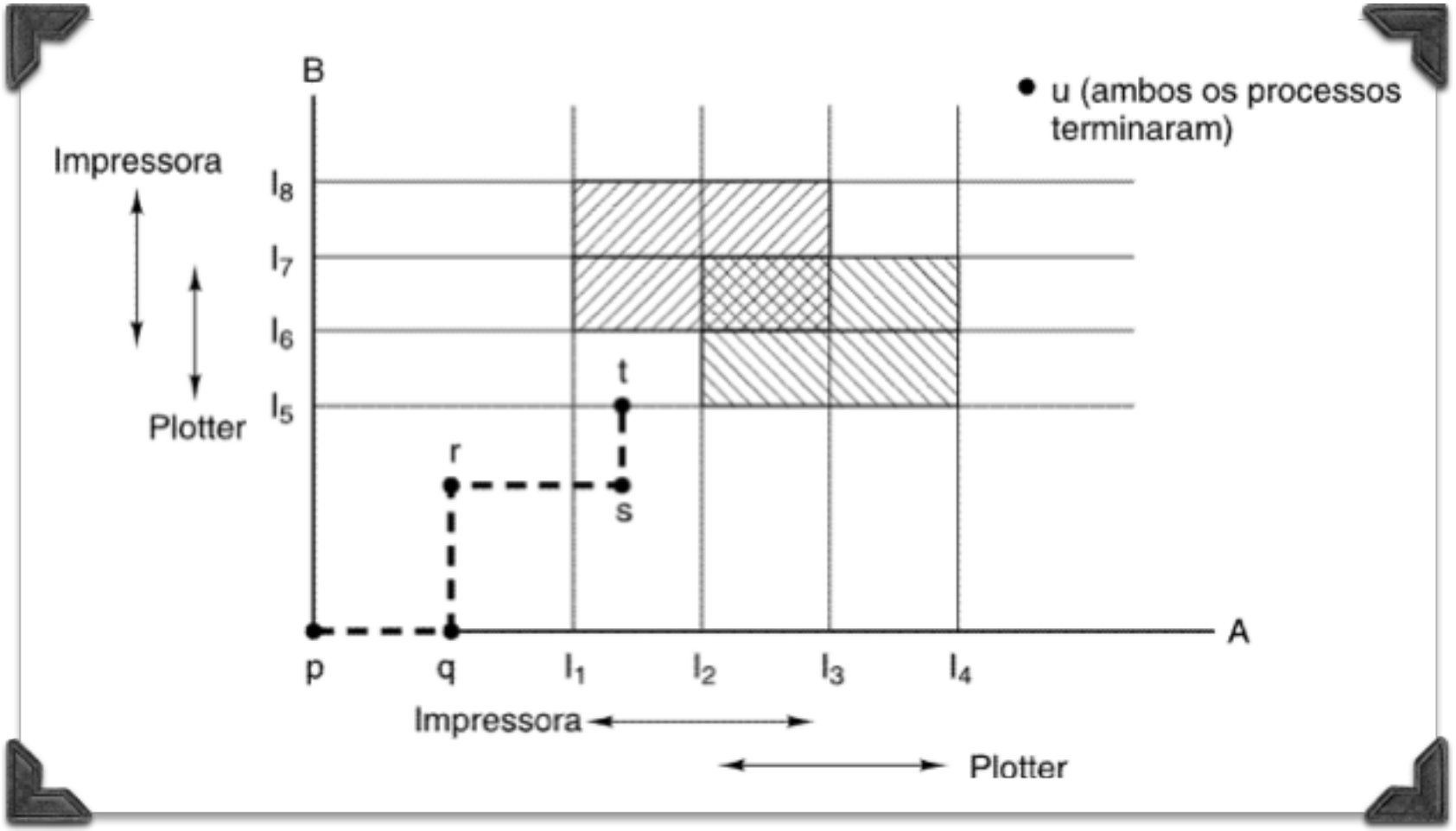
Detecção com 1 recurso de cada tipo



Recuperação de deadlock

- Recuperação através de preempção
  - retirar um recurso de algum outro processo
  - depende da natureza do recurso
- Recuperação através de restauração de estado
  - armazenar pontos de verificação de processos periodicamente
  - usa este estado salvo
  - restaura o processo se este é encontrado em estado de deadlock
- Recuperação através da eliminação de processos
  - forma mais grosseira mas mais simples de quebrar um deadlock
  - elimina um dos processos no ciclo de deadlock
  - os outros processos conseguem seus recursos
  - escolhe processo que pode ser reexecutado desde seu início

Evitando Deadlocks



Prevenção de deadlock

- Atacando a exclusão mútua
  - Recursos (como impressoras) podem fazer uso de spool
    - o daemon de impressão é o único que usa o recurso impressora
    - desta forma deadlock envolvendo a impressora é eliminado
  - Nem todos os dispositivos podem fazer uso de spool
  - Princípio:
    - evitar alocar um recurso quando ele não for absolutamente necessário
    - tentar assegurar que o menor número possível de processos possa de fato requisitar o recurso

Prevenção de Deadlock

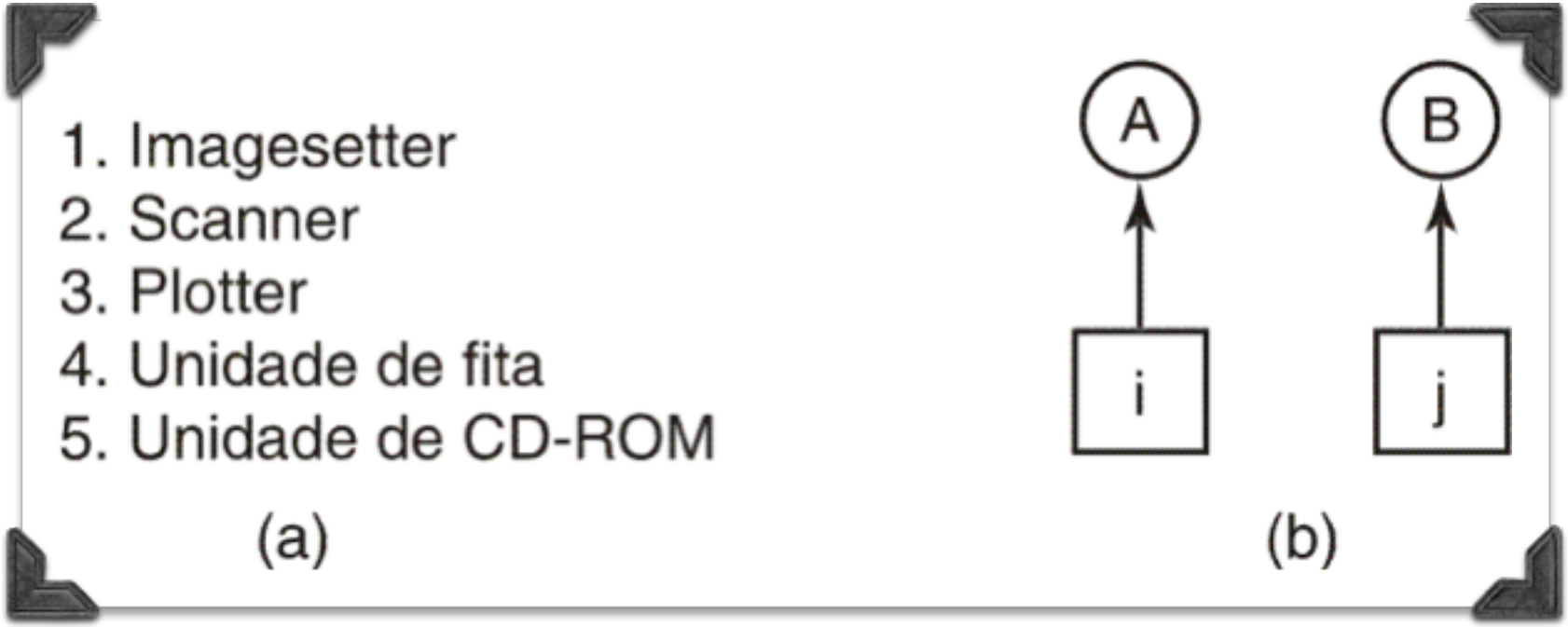
- Atacando a posse e espera
  - Exigir que todos os processos requisitem os recursos antes de iniciarem
    - um processo nunca tem que esperar por aquilo que precisa
  - Problemas
    - podem não saber quantos e quais recursos vão precisar no início da execução
    - e também retêm recursos que outros processos poderiam estar usando
  - Variação:
    - processo deve desistir de todos os recursos
    - para então requisitar todos os que são imediatamente necessários

Prevenção de Deadlock

- Atacando a condição de não-preempção
  - Esta é uma opção inviável
  - Considere um processo de posse de uma impressora
    - no meio da impressão
    - retoma a impressora a força

Prevenção de Deadlock

- Atacando a espera circular
  - Ordenação numérica



Prevenção de Deadlock

- Resumo

Condição	Abordagem contra deadlocks
Exclusão mútua	Usar spool em tudo
Posse-e-espera	Requisitar inicialmente todos os recursos necessários
Não preempção	Retomar os recursos alocados
Espera circular	Ordenar numericamente os recursos

Deadlocks sem envolvimento de recursos

- É possível que dois processos entrem em situação de deadlock
  - cada um espera que o outro faça algo
- Pode ocorrer com semáforos
  - cada processo executa um down() sobre dois semáforos (mutex e outro qualquer)
  - se executados na ordem errada, resulta em deadlock