

Aula 7

Professores:

Felipe M. G. França
Valmir C. Barbosa

Conteúdo:

Entrada/Saída

- Princípios de hardware e de software
- Impasses

Introdução

- Uma das principais tarefas do sistema operacional é a de gerenciar os dispositivos de E/S do computador:
 - Executar corretamente as operações de E/S.
 - Interceptar e tratar as interrupções geradas pelos dispositivos.
 - Tratar dos erros gerados ao executar as operações de E/S.
 - Prover uma interface abstrata para o acesso aos dispositivos, que seja simples, fácil de usar, e independente do dispositivo.
- O gerenciamento dos dispositivos é dividido em duas partes:
 - **Hardware de E/S:** como os dispositivos de E/S do computador funcionam internamente, e como são acessados pelo sistema.
 - **Software de E/S:** as partes do sistema operacional que tratam do gerenciamento dos dispositivos físicos do computador.

Introdução

- Os dispositivos de E/S são, em geral, classificados como ou dispositivos de **bloco**, ou de **caractere**:
- **Dispositivos de bloco**: as informações são armazenadas em um conjunto de blocos.
 - **Dispositivos de caractere**: as informações são passadas ao dispositivo por um fluxo seqüencial de caracteres.
 - Alguns dispositivos, como os temporizadores, não são nem dispositivos de bloco, nem dispositivos de caractere.



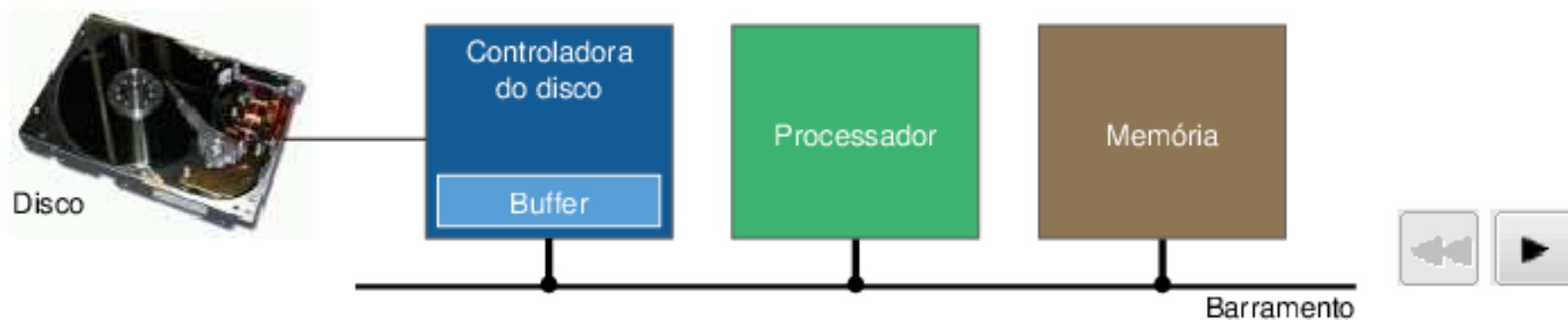
O drive de disquete é, como vimos, um dispositivo de bloco.



Uma impressora é, como vimos, um dispositivo de caractere.

Princípios de hardware de E/S

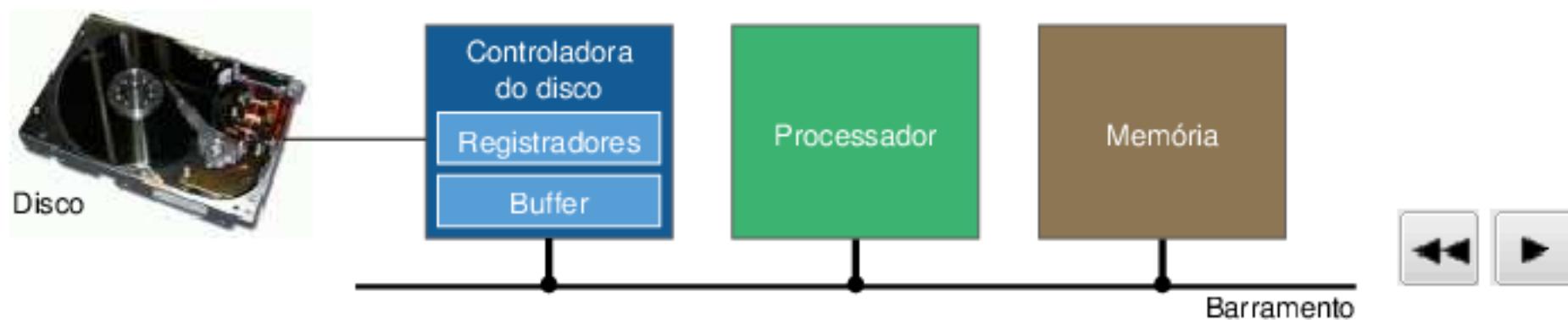
- ➡ Um dispositivo de E/S pode em geral ser dividido em duas partes, o que torna o projeto mais modular e genérico:
 - O **dispositivo físico**, que é a parte mecânica do dispositivo.
 - A **controladora ou adaptadora de dispositivo**, que é a parte eletrônica do dispositivo usada para acessar a parte mecânica.
- ➡ O sistema operacional executa as operações de E/S através da controladora do dispositivo:



O sistema operacional executa as operações de E/S através das controladoras do dispositivo. Cada controladora possui um conjunto de registradores usados para definir os comandos que podem ser executados pelo dispositivo, obter o estado do dispositivo, ou trocar dados com o dispositivo. Além disso, em geral, possuem um buffer para armazenamento temporário.

Princípios de hardware de E/S

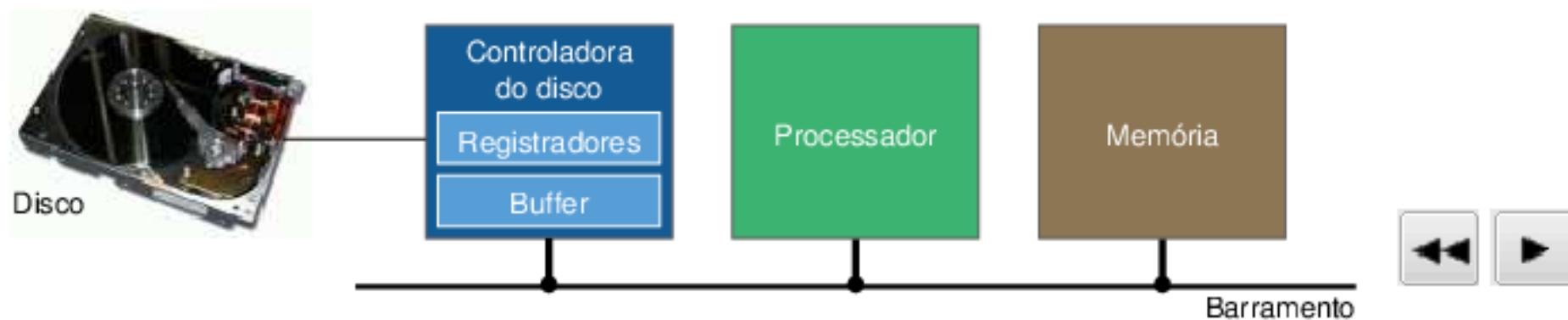
- ➡ Um dispositivo de E/S pode em geral ser dividido em duas partes, o que torna o projeto mais modular e genérico:
 - O **dispositivo físico**, que é a parte mecânica do dispositivo.
 - A **controladora ou adaptadora de dispositivo**, que é a parte eletrônica do dispositivo usada para acessar a parte mecânica.
- ➡ O sistema operacional executa as operações de E/S através da controladora do dispositivo:



Na **E/S mapeada em memória**, os registradores das controladoras estão na memória. O processador acessa os registradores usando instruções comuns para o acesso à memória. Já no acesso com **espaço de endereçamento especial**, os registradores estão na controladora, e são acessados através de instruções especiais de E/S.

Princípios de hardware de E/S

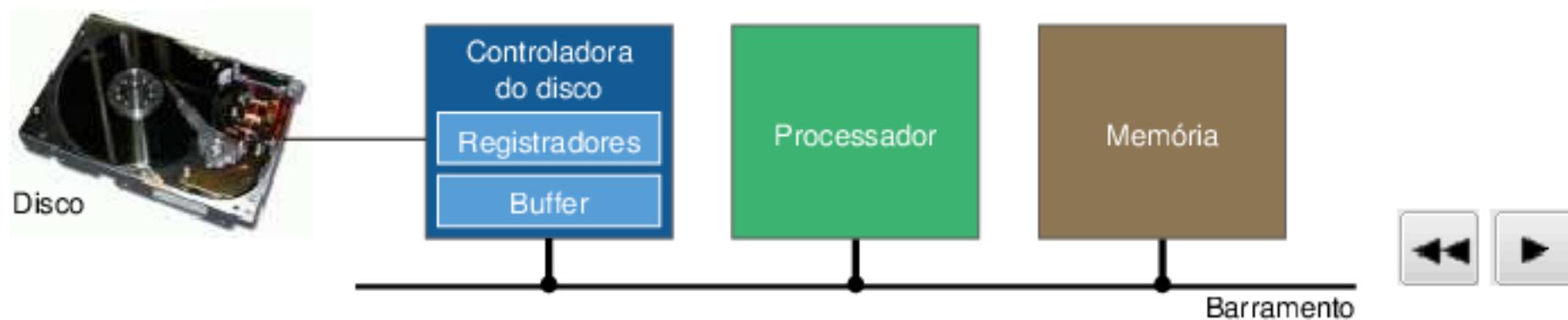
- ➡ Um dispositivo de E/S pode em geral ser dividido em duas partes, o que torna o projeto mais modular e genérico:
 - O **dispositivo físico**, que é a parte mecânica do dispositivo.
 - A **controladora ou adaptadora de dispositivo**, que é a parte eletrônica do dispositivo usada para acessar a parte mecânica.
- ➡ O sistema operacional executa as operações de E/S através da controladora do dispositivo:



Vamos supor que o sistema deseja ler dados do disco. Para poder executar a operação de E/S, o sistema a converte em um conjunto de comandos para a controladora, passados através dos seus registradores. Após a controladora receber os comandos, o processador poderá executar alguma outra tarefa.

Princípios de hardware de E/S

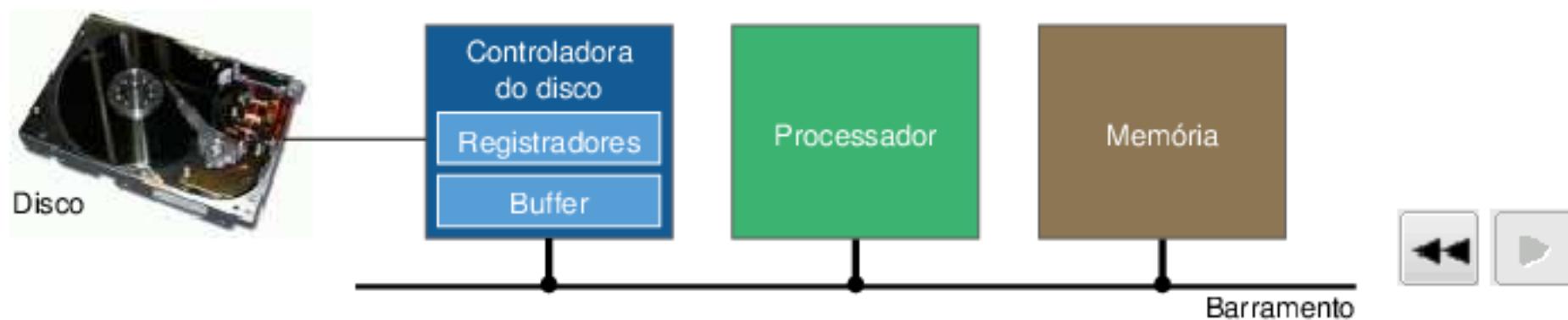
- ➡ Um dispositivo de E/S pode em geral ser dividido em duas partes, o que torna o projeto mais modular e genérico:
 - O **dispositivo físico**, que é a parte mecânica do dispositivo.
 - A **controladora ou adaptadora de dispositivo**, que é a parte eletrônica do dispositivo usada para acessar a parte mecânica.
- ➡ O sistema operacional executa as operações de E/S através da controladora do dispositivo:



A controladora envia os comandos ao dispositivo, que começa a transmitir os dados que foram pedidos. Depois disso, a controladora recebe os dados, faz as verificações necessárias para detectar os erros, e coloca estes dados no buffer. Depois de todos os dados serem recebidos, a controladora gera uma interrupção, para informar ao processador que a operação de E/S terminou.

Princípios de hardware de E/S

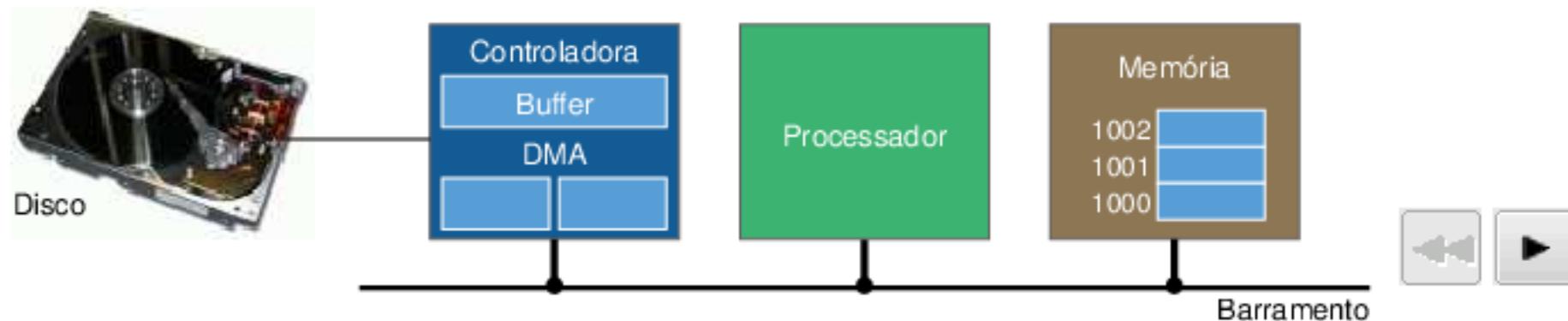
- ➡ Um dispositivo de E/S pode em geral ser dividido em duas partes, o que torna o projeto mais modular e genérico:
 - O **dispositivo físico**, que é a parte mecânica do dispositivo.
 - A **controladora ou adaptadora de dispositivo**, que é a parte eletrônica do dispositivo usada para acessar a parte mecânica.
- ➡ O sistema operacional executa as operações de E/S através da controladora do dispositivo:



Depois de receber a interrupção, o processador para de executar o processo atual, e chama uma rotina especial, o tratador da interrupção de disco, para tratar da interrupção gerada pelo disco. O processador então lê, usando os registradores da controladora, os dados lidos do disco armazenados no buffer, e os copia para a memória, além de verificar se ocorreu algum erro.

Princípios de hardware de E/S

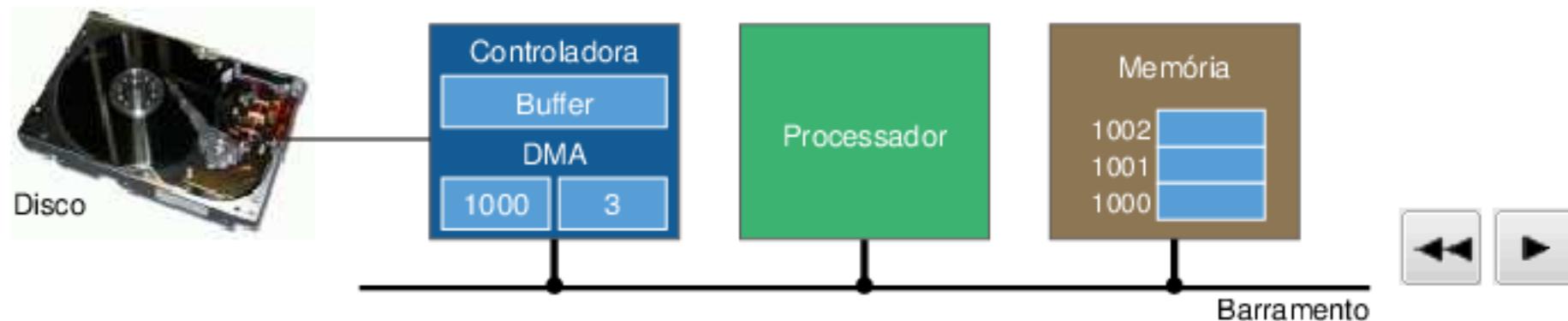
- ➡ Ao acessar um dispositivo, o processador deve ler (ou salvar) os dados no buffer interno da controladora da (na) memória.
- ➡ O acesso direto a memória, DMA (Direct Memory Access):
 - Permite à controladora ler (ou salvar) os dados enviados (ou recebidos) ao (pelo) dispositivo diretamente da (na) memória.
 - Não desperdiçamos mais tempo de processamento ao transferir dados entre a memória e a controladora.



Ao usarmos o DMA, o processador precisará somente enviar, à controladora, o endereço inicial da memória para o qual os dados lidos do disco devem ser copiados, e o número de bytes a serem copiados. Após executar os comandos enviados pelo processador e acessar o disco, a controladora copiará os dados para a memória, ao invés do processador, como na transparência anterior.

Princípios de hardware de E/S

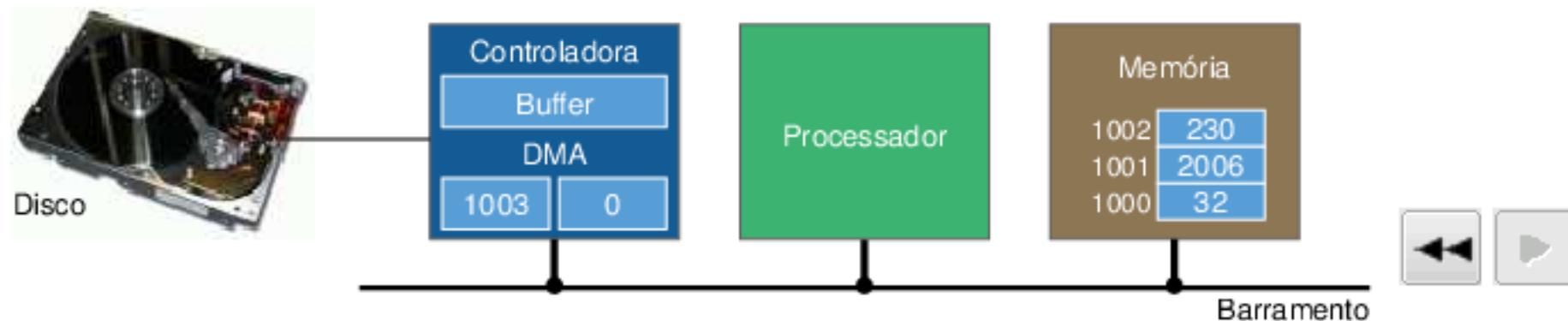
- ➡ Ao acessar um dispositivo, o processador deve ler (ou salvar) os dados no buffer interno da controladora da (na) memória.
- ➡ O acesso direto a memória, DMA (Direct Memory Access):
 - Permite à controladora ler (ou salvar) os dados enviados (ou recebidos) ao (pelo) dispositivo diretamente da (na) memória.
 - Não desperdiçamos mais tempo de processamento ao transferir dados entre a memória e a controladora.



O processador deseja ler 3 palavras do disco, e copiá-las a partir da posição de memória 1000. Então, ao enviar os comandos à controladora, o processador também envia o endereço da memória (1000), e o número de palavras (3).

Princípios de hardware de E/S

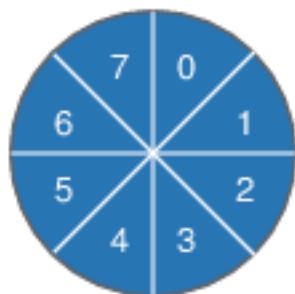
- ➡ Ao acessar um dispositivo, o processador deve ler (ou salvar) os dados no buffer interno da controladora da (na) memória.
- ➡ O acesso direto a memória, DMA (Direct Memory Access):
 - Permite à controladora ler (ou salvar) os dados enviados (ou recebidos) ao (pelo) dispositivo diretamente da (na) memória.
 - Não desperdiçamos mais tempo de processamento ao transferir dados entre a memória e a controladora.



Depois de a controladora ler os dados do disco para o seu buffer, ela começa a copiar os 3 dados para a memória, a partir do endereço 1000. Depois de terminar a cópia, a controladora gera uma interrupção para o processador, para informar que os dados já foram lidos e copiados para a memória.

Princípios de hardware de E/S

- ➡ Para algumas controladoras, o buffer interno é necessário, pois:
 - O dispositivo físico envia continuamente dados à controladora.
 - O barramento pode estar ocupado quando a controladora deseja acessar a memória.
 - A transferência de dados será abortada se a controladora não copiar um dado antes do dispositivo enviar o próximo dado.
- ➡ O processo de bufferização, usando o buffer interno e o DMA, tem um impacto importante no desempenho das operações de E/S:



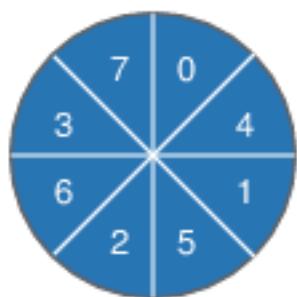
No caso do disco, se a controladora lê um bloco para o buffer, e leva o mesmo tempo gasto na leitura ao copiar o bloco para a memória, então, após a cópia, o próximo bloco já vai ter sido lido pelo disco, e com isso, será necessário esperar que este bloco seja lido novamente (depois de o disco ler 7 blocos).



- ➡ Para o uso do DMA ser vantajoso, o processador não pode ficar esperando pelo término da transferência dos dados.

Princípios de hardware de E/S

- ➡ Para algumas controladoras, o buffer interno é necessário, pois:
 - O dispositivo físico envia continuamente dados à controladora.
 - O barramento pode estar ocupado quando a controladora deseja acessar a memória.
 - A transferência de dados será abortada se a controladora não copiar um dado antes do dispositivo enviar o próximo dado.
- ➡ O processo de bufferização, usando o buffer interno e o DMA, tem um impacto importante no desempenho das operações de E/S:



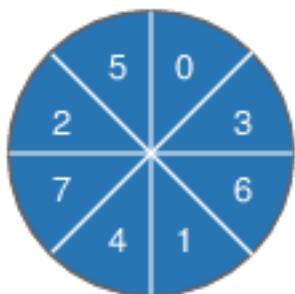
Se a controladora leva o mesmo tempo para ler e copiar um bloco do disco, então podemos numerar os blocos de acordo com a **intercalação única**, em que sempre existirá pelo menos um bloco entre dois blocos com numeração consecutiva. Com isso, agora a controladora poderá ler os blocos numerados de modo consecutivo, pois nunca perderemos uma leitura.



- ➡ Para o uso do DMA ser vantajoso, o processador não pode ficar esperando pelo término da transferência dos dados.

Princípios de hardware de E/S

- ➡ Para algumas controladoras, o buffer interno é necessário, pois:
 - O dispositivo físico envia continuamente dados à controladora.
 - O barramento pode estar ocupado quando a controladora deseja acessar a memória.
 - A transferência de dados será abortada se a controladora não copiar um dado antes do dispositivo enviar o próximo dado.
- ➡ O processo de bufferização, usando o buffer interno e o DMA, tem um impacto importante no desempenho das operações de E/S:



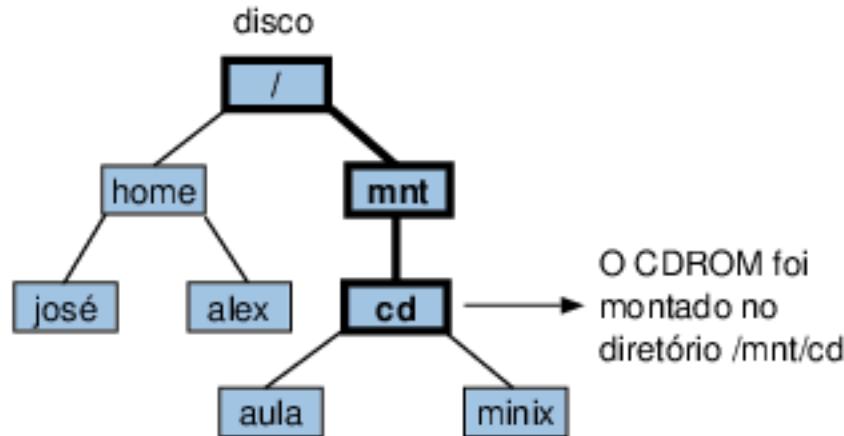
Se a controladora demorar mais, poderemos aumentar o fator de intercalação, colocando um certo número de blocos entre dois blocos consecutivos. No exemplo vemos a **intercalação dupla**, usada quando a controladora leva, para copiar o bloco, o dobro do tempo que o disco gasta para ler o bloco.



- ➡ Para o uso do DMA ser vantajoso, o processador não pode ficar esperando pelo término da transferência dos dados.

Princípios de software de E/S

- ➔ Um software de E/S deve ser projetado para atingirmos a meta da **independência de dispositivo**:
 - As operações executadas em um dispositivo físico devem independer de como este funciona.
 - Os detalhes do funcionamento dos dispositivos físicos são ocultados e tratados pelo sistema operacional.
- ➔ Um software de E/S deve definir uma **atribuição uniforme de nomes** aos dispositivos físicos:
 - Os nomes devem ser uma string ou um número inteiro.
 - O significado dos nomes deve independe dos dispositivos.



Se o drive de CDROM foi montado no diretório /mnt/cd, então, se usarmos o programa sort, este será capaz de ordenar tanto o arquivo alex, que está no disco, como o arquivo aula, que está no CD, pois o acesso a ambos os arquivos é feito via nome de caminho.

Princípios de software de E/S

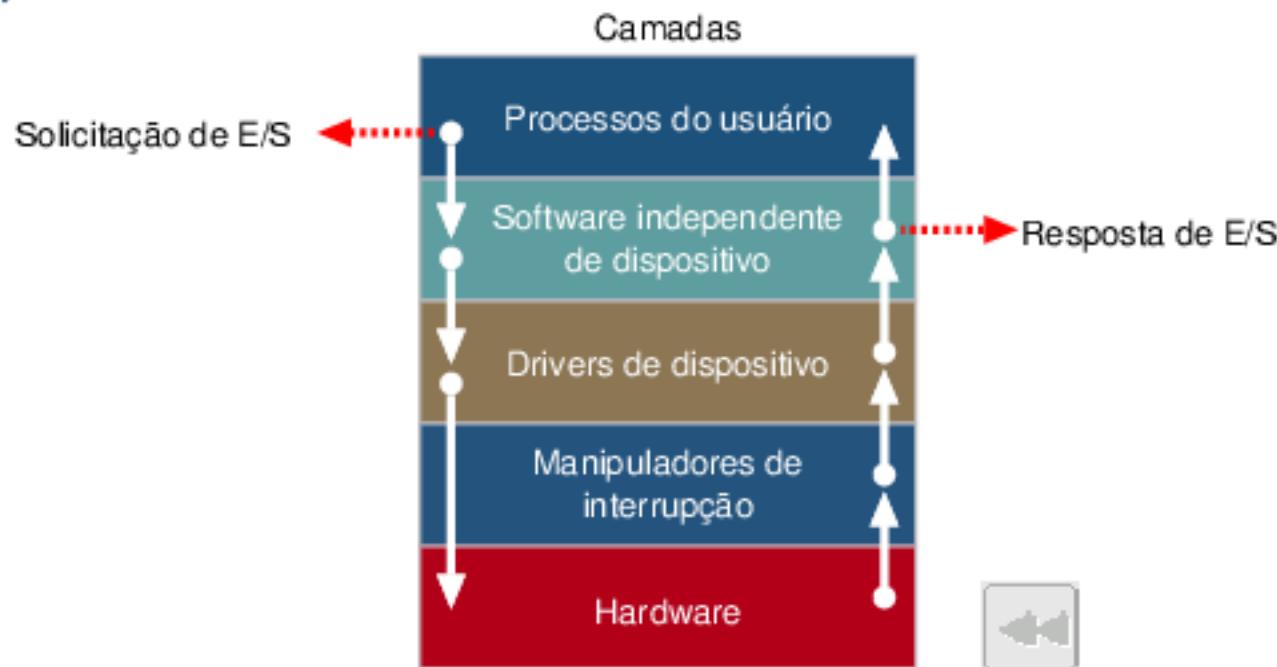
- ➡ Um detalhe importante é o da transferência dos dados:
 - **Transferências síncronas:** bloqueiam até a operação de E/S ser finalizada.
 - **Transferências assíncronas:** não bloqueiam, pois uma interrupção é gerada quando a operação é finalizada.
- ➡ Como a maior parte da E/S é assíncrona, o sistema deve tratar de bloquear o processo, e desbloqueá-lo ao ocorrer a interrupção.
- ➡ Um outro detalhe é o do uso dos dispositivos físicos:
 - **Dispositivos compartilháveis:** vários processos de usuário podem usar simultaneamente o dispositivo.
 - Um exemplo seria o disco rígido do computador.
 - **Dispositivos dedicados:** somente um processo de usuário pode usar o dispositivo por um certo tempo.
 - Um exemplo seria uma unidade de fita usada para backup.

Princípios de software de E/S

- ➡ O software de E/S é organizado como uma hierarquia em camadas:
 - As camadas mais baixas tratam de ocultar os detalhes do funcionamento dos dispositivos físicos.
 - As camadas mais altas tratam de disponibilizar, aos processos dos usuários, uma interface de acesso simples e fácil de usar.
- ➡ Uma questão importante é a do tratamento dos erros:
 - Os erros devem ser tratados, se possível, pela camada mais próxima possível ao hardware:
 - Por exemplo, os erros ao acessar o disco do computador devem ser resolvidos, se possível, pelo driver deste disco.
 - Uma camada somente deve saber da existência de um erro se as camadas abaixo desta não puderam tratar deste erro.

Princípios de software de E/S

→ Uma possível hierarquia para E/S:



Na figura vemos a camada de hardware, e a divisão do software de E/S em quatro camadas, onde as inferiores tratam do acesso e do gerenciamento dos dispositivos, e as superiores tratam das operações de E/S executadas em dispositivos abstratos. Pressione o mouse sobre cada camada para ver uma descrição resumida da sua função.

Recursos

- Um **recurso** é ou um dispositivo físico (dedicado) do hardware, ou um conjunto de informações, que deve ser exclusivamente usado.

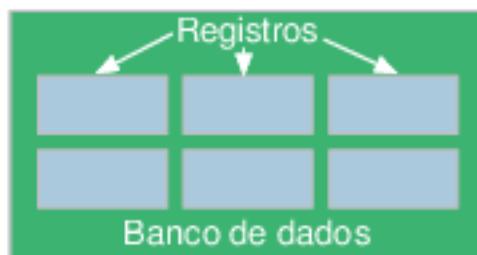


A impressora é um recurso, pois é um dispositivo dedicado, devido ao fato de somente um processo poder usá-la em um dado intervalo de tempo.

- Um processo pode solicitar vários recursos, inclusive várias cópias do mesmo recurso, e pode usar qualquer cópia de um recurso.
- Quando desejar usar um recurso, um processo deverá:
- **Solicitar o recurso:** esperar pelo recurso, até obtê-lo.
 - **Usar o recurso:** fazer o que for necessário com o recurso.
 - **Liberar o recurso:** devolver o controle do recurso ao sistema.

Recursos

- Um **recurso** é ou um dispositivo físico (dedicado) do hardware, ou um conjunto de informações, que deve ser exclusivamente usado.



Um outro exemplo de um recurso são os registros em uma base de dados. Se um processo deseja alterar o conteúdo de um ou mais registros, este processo deve bloqueá-los, para evitar que outros processos tentem acessar estes registros.

- Um processo pode solicitar vários recursos, inclusive várias cópias do mesmo recurso, e pode usar qualquer cópia de um recurso.
- Quando desejar usar um recurso, um processo deverá:
- **Solicitar o recurso:** esperar pelo recurso, até obtê-lo.
 - **Usar o recurso:** fazer o que for necessário com o recurso.
 - **Liberar o recurso:** devolver o controle do recurso ao sistema.

Recursos



Existem dois tipos de recursos no sistema:

- **Preemptíveis:** pode ser tirado do processo que o possui sem prejudicar o resultado da computação.



A memória é um exemplo de um recurso preemptivo. No exemplo, o processo A está usando a memória.

- **Não-preemptíveis:** se o recurso for tirado do processo antes de este liberá-lo, o resultado da computação será incorreto.



Fluxo de caracteres (A)

a	r	q	u	i	v	o	s
---	---	---	---	---	---	---	---

Fluxo de caracteres (B)

s	i	s	t	e	m	a	s
---	---	---	---	---	---	---	---



Folha

arquivos
sistemas

A impressora é um dispositivo dedicado, e por isso, é um recurso não-preemptivo. Se os processos A e B imprimem as suas saídas em uma folha, então se A imprimir a saída, liberar a impressora, e B imprimir a sua saída, o resultado será o esperado.

Recursos

→ Existem dois tipos de recursos no sistema:

- **Preemptíveis:** pode ser tirado do processo que o possui sem prejudicar o resultado da computação.



Depois de algum tempo, um processo B, que usa toda a memória, começa a sua execução. O recurso será tirado de A e passado a B. Depois de algum tempo, A obterá novamente o recurso, e poderá continuar a computação de onde parou.

- **Não-preemptíveis:** se o recurso for tirado do processo antes de este liberá-lo, o resultado da computação será incorreto.



A impressora é um dispositivo dedicado, e por isso, é um recurso não-preemptivo. Se os processos A e B imprimem as suas saídas em uma folha, então se A imprimir a saída, liberar a impressora, e B imprimir a sua saída, o resultado será o esperado.

Impasses

- ➔ Um conjunto de processos está em um estado de **impasse** se:
 - Cada um dos processos está esperando por um evento que pode ser gerado somente por um outro processo do conjunto.

Processo A



Processo B

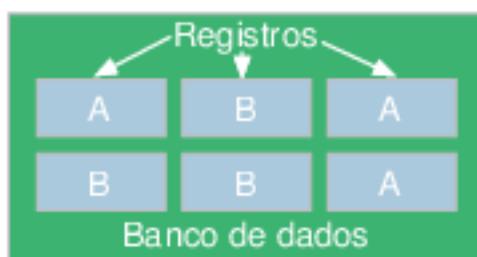


Se o processo A possui a impressora, e o processo B possui o CDROM, e se A desejar usar o CDROM, e B desejar usar a impressora, sem liberarem os recursos que possuem, então estes processos esperarão eternamente pelos recursos.

- ➔ A tarefa executada pelos processos cooperativos nunca será completada, pois todos os processos estarão esperando.
- ➔ Os impasses em geral envolvem recursos não-preemptivos, pois:
 - Se um recurso é preemptivo, os impasses podem ser evitados através da realocação dos recursos aos processos.
 - Isso não pode ser feito com os recursos não-preemptivos, pois se não o resultado da tarefa cooperativa seria incorreto.

Impasses

- ➔ Um conjunto de processos está em um estado de **impasse** se:
 - Cada um dos processos está esperando por um evento que pode ser gerado somente por um outro processo do conjunto.



Imagine que os processos A e B bloquearam alguns dos registros de um banco de dados. Se A desejar usar um dos registros de B, e B desejar usar um dos registros de A, sem desbloquearem antes os seus registros, então os processos A e B esperarão para sempre pelos novos registros.

- ➔ A tarefa executada pelos processos cooperativos nunca será completada, pois todos os processos estarão esperando.
- ➔ Os impasses em geral envolvem recursos não-preemptivos, pois:
 - Se um recurso é preemptivo, os impasses podem ser evitados através da realocação dos recursos aos processos.
 - Isso não pode ser feito com os recursos não-preemptivos, pois se não o resultado da tarefa cooperativa seria incorreto.

Impasses

→ Um impasse ocorrerá somente se existirem as seguintes condições:

- **Condição de exclusão mútua**: cada recurso está atribuído a um único processo em um dado intervalo de tempo.
- **Condição de segura e espera**: um processo pode solicitar novos recursos quando ainda está segurando outros recursos.
- **Condição de nenhuma preempção**: um recurso concedido a um processo somente pode ser liberado pelo processo.
- **Condição de espera circular**: existe uma cadeia circular de dependência entre os processos.

Processo A



Processo B



Processo C



No exemplo, os processos A, B, e C possuem, respectivamente, a impressora, o CDROM, e a unidade de fita. Teremos uma dependência circular se A solicitar o uso do CDROM, B o uso da unidade de fita, e C o uso da impressora, sem liberarem os recursos que possuem.

Modelagem dos impasses

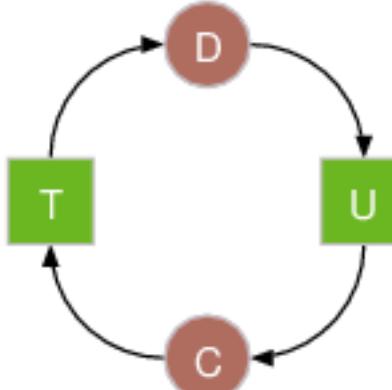
- Podemos usar o seguinte **grafo de recursos**, para modelar as solicitações e liberações dos recursos pelos processos.
- Existe um nó para cada um dos processos (os círculos).
 - Existe um nó para cada um dos recursos (os quadrados).
 - Se um recurso está alocado a um processo, existe um arco do nó deste recurso para o nó deste processo.
 - Se um processo fez uma solicitação a um recurso, existirá um arco do nó deste processo para o nó deste recurso.



O processo A
possui o recurso R



O processo B
deseja o recurso S



Um estado de impasse,
representado por um ciclo.

Neste caso, os processos C e D, possuem, respectivamente, os recursos U e T. Existe um estado de impasse porque C solicitou o recurso T, e D o recurso U, sem liberarem os recursos que possuem.

Modelagem dos impasses

- Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.
 - Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:

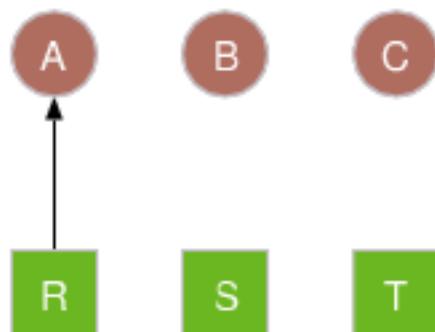


Vamos ver agora como a ordem das solicitações pode ou não levar a um impasse.



Modelagem dos impasses

- Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.
- Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:



Se tivermos a seguinte seqüência de solicitações, sem nenhuma liberação de recursos entre estas solicitações, teremos um impasse, pois o grafo terá o ciclo R-A-S-B-T-C-R:

A solicita e obtém R.

B solicita e obtém S.

C solicita e obtém T.

A solicita S e bloqueia.

B solicita T e bloqueia.

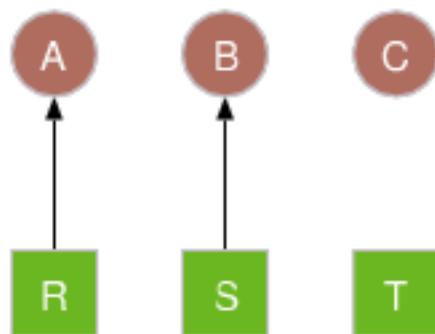
C solicita R e bloqueia.



Modelagem dos impasses

→ Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.

- Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:



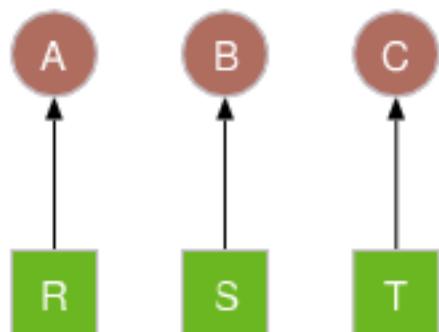
Se tivermos a seguinte seqüência de solicitações, sem nenhuma liberação de recursos entre estas solicitações, teremos um impasse, pois o grafo terá o ciclo R-A-S-B-T-C-R:

- A solicita e obtém R.
- B solicita e obtém S.**
- C solicita e obtém T.
- A solicita S e bloqueia.
- B solicita T e bloqueia.
- C solicita R e bloqueia.



Modelagem dos impasses

- Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.
 - Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:



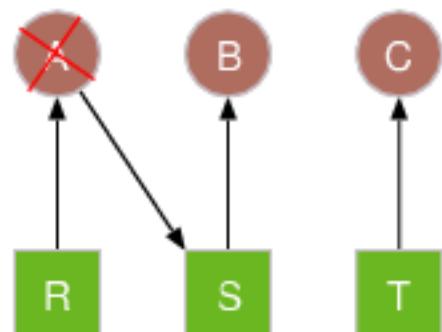
Se tivermos a seguinte seqüência de solicitações, sem nenhuma liberação de recursos entre estas solicitações, teremos um impasse, pois o grafo terá o ciclo R-A-S-B-T-C-R:

- A solicita e obtém R.
- B solicita e obtém S.
- C solicita e obtém T.**
- A solicita S e bloqueia.
- B solicita T e bloqueia.
- C solicita R e bloqueia.



Modelagem dos impasses

- Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.
 - Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:



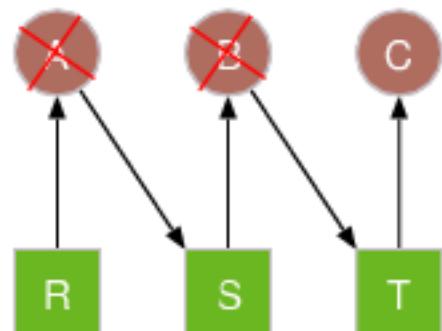
Se tivermos a seguinte seqüência de solicitações, sem nenhuma liberação de recursos entre estas solicitações, teremos um impasse, pois o grafo terá o ciclo R-A-S-B-T-C-R:

- A solicita e obtém R.
- B solicita e obtém S.
- C solicita e obtém T.
- A solicita S e bloqueia.**
- B solicita T e bloqueia.
- C solicita R e bloqueia.



Modelagem dos impasses

- Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.
 - Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:



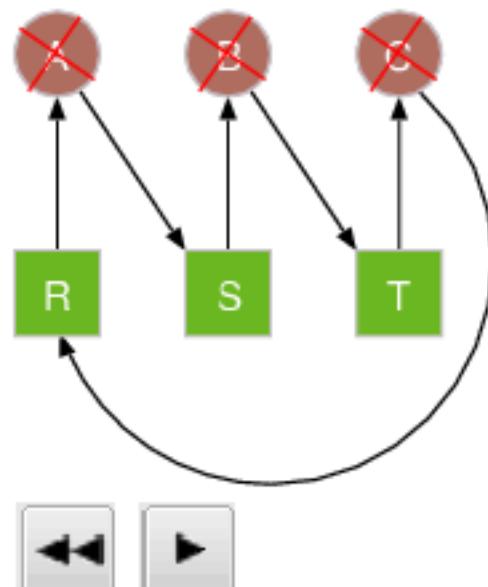
Se tivermos a seguinte seqüência de solicitações, sem nenhuma liberação de recursos entre estas solicitações, teremos um impasse, pois o grafo terá o ciclo R-A-S-B-T-C-R:

- A solicita e obtém R.
- B solicita e obtém S.
- C solicita e obtém T.
- A solicita S e bloqueia.
- B solicita T e bloqueia.**
- C solicita R e bloqueia.



Modelagem dos impasses

- Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.
 - Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:

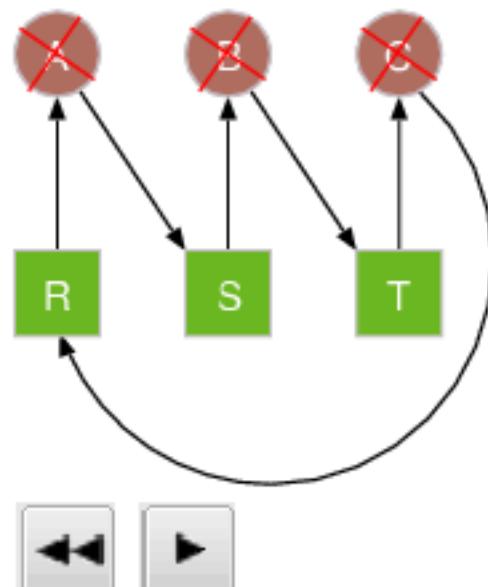


Se tivermos a seguinte seqüência de solicitações, sem nenhuma liberação de recursos entre estas solicitações, teremos um impasse, pois o grafo terá o ciclo R-A-S-B-T-C-R:

- A solicita e obtém R.
- B solicita e obtém S.
- C solicita e obtém T.
- A solicita S e bloqueia.
- B solicita T e bloqueia.
- C solicita R e bloqueia.**

Modelagem dos impasses

- Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.
 - Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:



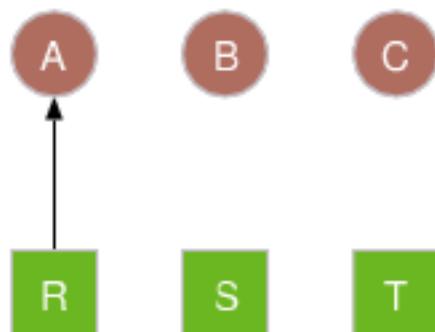
Se tivermos a seguinte seqüência de solicitações, sem nenhuma liberação de recursos entre estas solicitações, teremos um impasse, pois o grafo terá o **ciclo R-A-S-B-T-C-R**:

- A solicita e obtém R.
- B solicita e obtém S.
- C solicita e obtém T.
- A solicita S e bloqueia.
- B solicita T e bloqueia.
- C solicita R e bloqueia.

Modelagem dos impasses

→ Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.

- Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:



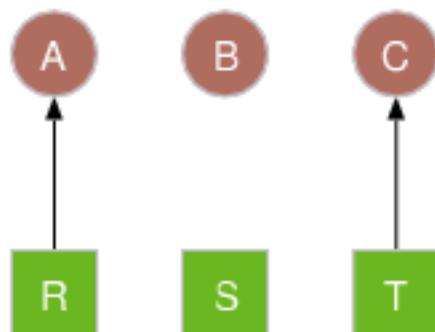
Se, na lista anterior, as solicitações de um dos processos, por exemplo, o B, forem postergadas para depois de A liberar os recursos, o impasse não irá mais ocorrer. Logo, a seqüência de solicitações e de liberações dadas a seguir não geram um impasse:

- A solicita e obtém R.
- C solicita e obtém T.
- A solicita e obtém S.
- C solicita R e bloqueia.
- A libera R e depois S.
- C é desbloqueado e obtém R.
- B solicita e obtém S.
- B solicita T e bloqueia.
- C libera T e R.
- B é desbloqueado e obtém T.
- B libera S e T.



Modelagem dos impasses

- Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.
- Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:

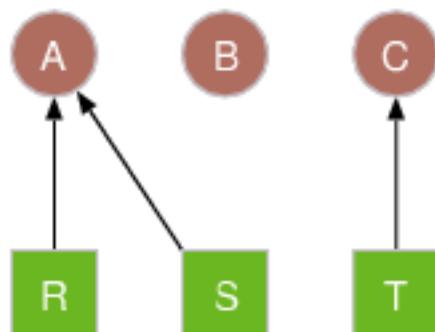


Se, na lista anterior, as solicitações de um dos processos, por exemplo, o B, forem postergadas para depois de A liberar os recursos, o impasse não irá mais ocorrer. Logo, a seqüência de solicitações e de liberações dadas a seguir não geram um impasse:

- A solicita e obtém R.
- C solicita e obtém T.**
- A solicita e obtém S.
- C solicita R e bloqueia.
- A libera R e depois S.
- C é desbloqueado e obtém R.
- B solicita e obtém S.
- B solicita T e bloqueia.
- C libera T e R.
- B é desbloqueado e obtém T.
- B libera S e T.

Modelagem dos impasses

- Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.
 - Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:



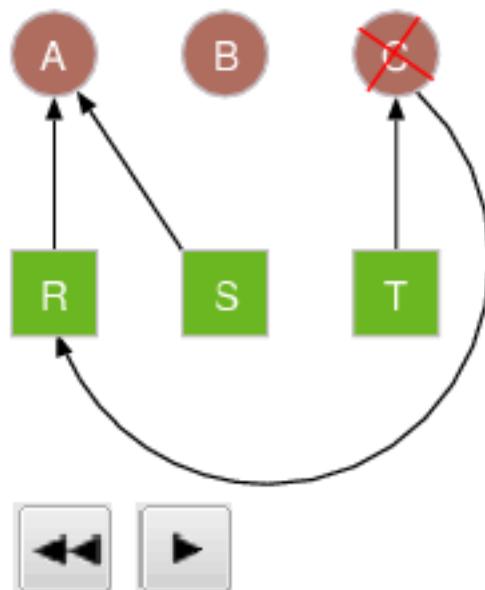
Se, na lista anterior, as solicitações de um dos processos, por exemplo, o B, forem postergadas para depois de A liberar os recursos, o impasse não irá mais ocorrer. Logo, a seqüência de solicitações e de liberações dadas a seguir não geram um impasse:

- A solicita e obtém R.
- C solicita e obtém T.
- A solicita e obtém S.**
- C solicita R e bloqueia.
- A libera R e depois S.
- C é desbloqueado e obtém R.
- B solicita e obtém S.
- B solicita T e bloqueia.
- C libera T e R.
- B é desbloqueado e obtém T.
- B libera S e T.



Modelagem dos impasses

- Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.
 - Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:

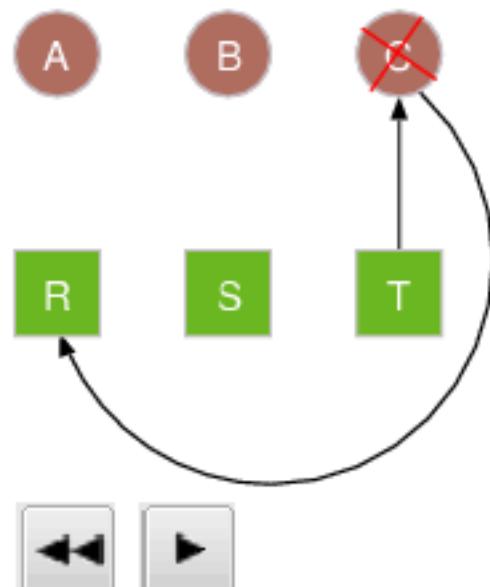


Se, na lista anterior, as solicitações de um dos processos, por exemplo, o B, forem postergadas para depois de A liberar os recursos, o impasse não irá mais ocorrer. Logo, a seqüência de solicitações e de liberações dadas a seguir não geram um impasse:

- A solicita e obtém R.
- C solicita e obtém T.
- A solicita e obtém S.
- C solicita R e bloqueia.**
- A libera R e depois S.
- C é desbloqueado e obtém R.
- B solicita e obtém S.
- B solicita T e bloqueia.
- C libera T e R.
- B é desbloqueado e obtém T.
- B libera S e T.

Modelagem dos impasses

- Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.
- Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:

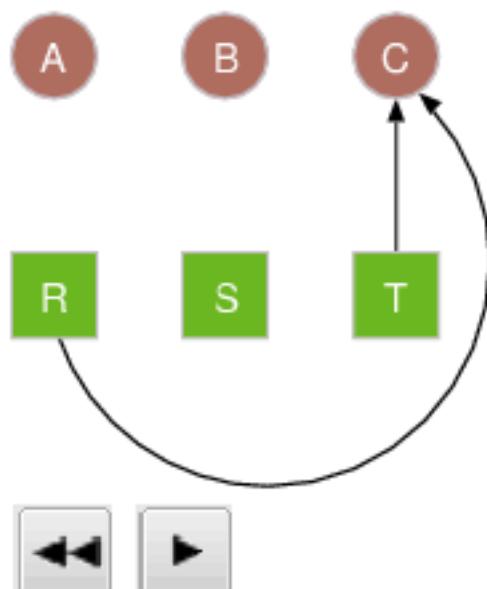


Se, na lista anterior, as solicitações de um dos processos, por exemplo, o B, forem postergadas para depois de A liberar os recursos, o impasse não irá mais ocorrer. Logo, a seqüência de solicitações e de liberações dadas a seguir não geram um impasse:

- A solicita e obtém R.
- C solicita e obtém T.
- A solicita e obtém S.
- C solicita R e bloqueia.
- A libera R e depois S.**
- C é desbloqueado e obtém R.
- B solicita e obtém S.
- B solicita T e bloqueia.
- C libera T e R.
- B é desbloqueado e obtém T.
- B libera S e T.

Modelagem dos impasses

- Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.
 - Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:

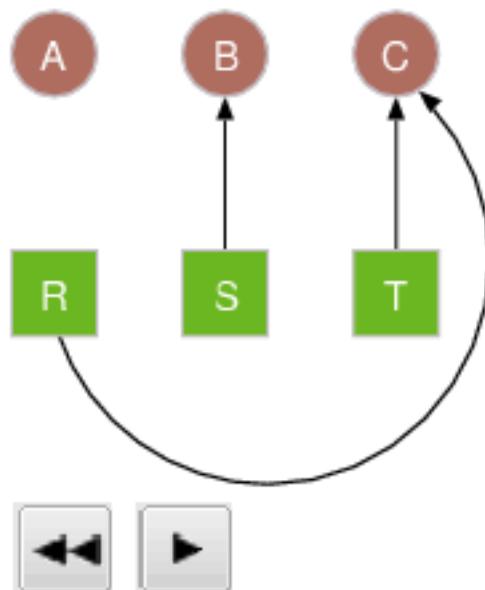


Se, na lista anterior, as solicitações de um dos processos, por exemplo, o B, forem postergadas para depois de A liberar os recursos, o impasse não irá mais ocorrer. Logo, a seqüência de solicitações e de liberações dadas a seguir não geram um impasse:

- A solicita e obtém R.
- C solicita e obtém T.
- A solicita e obtém S.
- C solicita R e bloqueia.
- A libera R e depois S.
- C é desbloqueado e obtém R.**
- B solicita e obtém S.
- B solicita T e bloqueia.
- C libera T e R.
- B é desbloqueado e obtém T.
- B libera S e T.

Modelagem dos impasses

- Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.
 - Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:

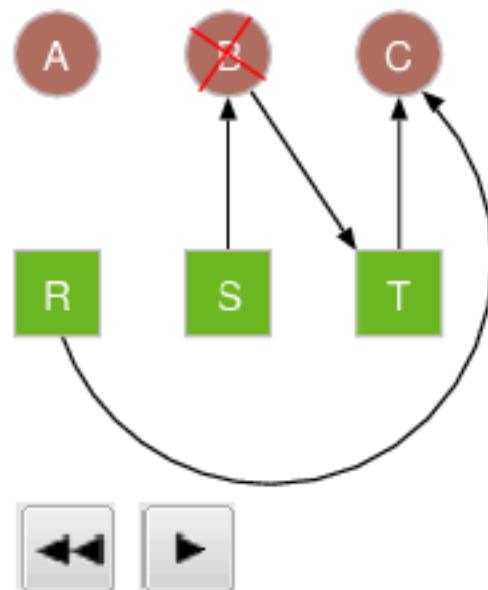


Se, na lista anterior, as solicitações de um dos processos, por exemplo, o B, forem postergadas para depois de A liberar os recursos, o impasse não irá mais ocorrer. Logo, a seqüência de solicitações e de liberações dadas a seguir não geram um impasse:

- A solicita e obtém R.
- C solicita e obtém T.
- A solicita e obtém S.
- C solicita R e bloqueia.
- A libera R e depois S.
- C é desbloqueado e obtém R.
- B solicita e obtém S.**
- B solicita T e bloqueia.
- C libera T e R.
- B é desbloqueado e obtém T.
- B libera S e T.

Modelagem dos impasses

- Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.
 - Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:

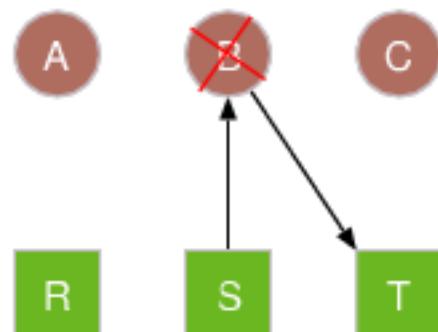


Se, na lista anterior, as solicitações de um dos processos, por exemplo, o B, forem postergadas para depois de A liberar os recursos, o impasse não irá mais ocorrer. Logo, a seqüência de solicitações e de liberações dadas a seguir não geram um impasse:

- A solicita e obtém R.
- C solicita e obtém T.
- A solicita e obtém S.
- C solicita R e bloqueia.
- A libera R e depois S.
- C é desbloqueado e obtém R.
- B solicita e obtém S.
- B solicita T e bloqueia.**
- C libera T e R.
- B é desbloqueado e obtém T.
- B libera S e T.

Modelagem dos impasses

- Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.
- Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:

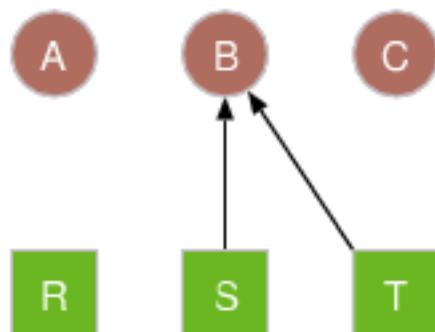


Se, na lista anterior, as solicitações de um dos processos, por exemplo, o B, forem postergadas para depois de A liberar os recursos, o impasse não irá mais ocorrer. Logo, a seqüência de solicitações e de liberações dadas a seguir não geram um impasse:

- A solicita e obtém R.
- C solicita e obtém T.
- A solicita e obtém S.
- C solicita R e bloqueia.
- A libera R e depois S.
- C é desbloqueado e obtém R.
- B solicita e obtém S
- B solicita T e bloqueia.
- C libera T e R.**
- B é desbloqueado e obtém T.
- B libera S e T.

Modelagem dos impasses

- Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.
- Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:



Se, na lista anterior, as solicitações de um dos processos, por exemplo, o B, forem postergadas para depois de A liberar os recursos, o impasse não irá mais ocorrer. Logo, a seqüência de solicitações e de liberações dadas a seguir não geram um impasse:

- A solicita e obtém R.
- C solicita e obtém T.
- A solicita e obtém S.
- C solicita R e bloqueia.
- A libera R e depois S.
- C é desbloqueado e obtém R.
- B solicita e obtém S
- B solicita T e bloqueia.
- C libera T e R.
- B é desbloqueado e obtém T.**
- B libera S e T.

Modelagem dos impasses

→ Vamos ver um exemplo prático da utilidade do grafo de recursos, para três processos A, B, e C, que usam três recursos R, S, e T.

- Dependendo da ordem em que as solicitações e liberações forem feitas, poderemos ter ou não um impasse:



Se, na lista anterior, as solicitações de um dos processos, por exemplo, o B, forem postergadas para depois de A liberar os recursos, o impasse não irá mais ocorrer. Logo, a seqüência de solicitações e de liberações dadas a seguir não geram um impasse:



- A solicita e obtém R.
- C solicita e obtém T.
- A solicita e obtém S.
- C solicita R e bloqueia.
- A libera R e depois S.
- C é desbloqueado e obtém R.
- B solicita e obtém S
- B solicita T e bloqueia.
- C libera T e R.
- B é desbloqueado e obtém T.
- B libera S e T.



Modelagem dos impasses

- ➡ O grafo que modela o compartilhamento dos recursos:
 - Pode ser usado para detectar se uma dada seqüência de solicitações e de liberações gera ou não um impasse.
 - Pode ser usado pelo sistema operacional para tentar evitar a ocorrência dos impasses.
 - Pode ser facilmente estendido para vários recursos de um mesmo tipo.
- ➡ Podemos usar quatro estratégias para tratar dos impasses:
 - Ignorar totalmente a existência dos impasses.
 - Detectar o impasse e recuperar o sistema após a ocorrência deste impasse.
 - Evitar a ocorrência dos impasses em tempo de execução, ao alojar os recursos aos processos.
 - Impedir ocorrência de impasses, definindo regras que impedem a existência de uma das quatro condições necessárias.

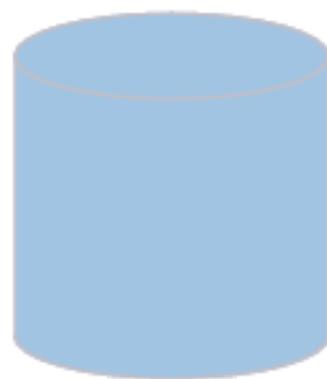
Detecção e recuperação

- ➡ O sistema usa o grafo de recursos para detectar os impasses e recuperar o sistema deste erro:
 - O sistema atualiza o grafo a cada solicitação ou liberação de um recurso.
 - Se o sistema detectar um ciclo no grafo:
 - Um processo do ciclo, escolhido aleatoriamente, é terminado, e os seus recursos são liberados.
 - A escolha continua até que o grafo seja acíclico.
 - O problema é que nem sempre é possível reiniciar um processo.
 - Um outro problema é o de reverter todas as alterações feitas durante a execução de cada um dos processos terminados.
- ➡ Uma outra idéia, mais simples, é a de eliminar um processo que esteja bloqueado por um longo período de tempo.
- ➡ Esta estratégia é em geral usada nos sistemas de lote dos computadores de grande porte.

Prevenção de impasses

- ➡ A idéia é evitar a ocorrência dos impasses garantindo que uma das quatro condições necessárias nunca ocorram.
- ➡ Para evitar a condição da exclusão mútua:
 - Nenhum dos recursos é dedicado.
 - Como alguns recursos devem ser dedicados, deveremos definir um outro critério para garantir o uso exclusivo destes recursos.

Disco



Espaço disponível



Para alguns recursos que são dedicados, como a impressora, podemos definir que somente um processo, o spoller, solicita a impressora. Para um processo imprimir a sua saída, este deverá criar um arquivo com esta saída, e colocá-lo em um diretório especial, o diretório de spoller.

Prevenção de impasses

- ➡ A idéia é evitar a ocorrência dos impasses garantindo que uma das quatro condições necessárias nunca ocorram.
- ➡ Para evitar a condição da exclusão mútua:
 - Nenhum dos recursos é dedicado.
 - Como alguns recursos devem ser dedicados, deveremos definir um outro critério para garantir o uso exclusivo destes recursos.

Disco

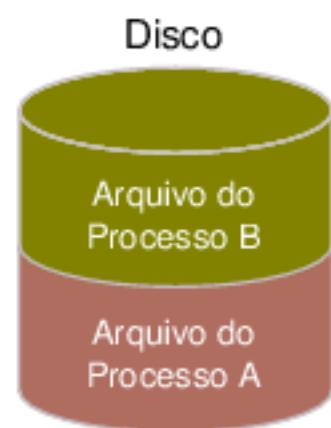


Suponha que o processo A deseja imprimir a sua saída. Logo, A deverá criar um arquivo, e gravar toda a sua saída nele. Suponha que A parou de gravar no arquivo, pois começou a executar uma computação, e suponha que a computação é demorada.



Prevenção de impasses

- ➡ A idéia é evitar a ocorrência dos impasses garantindo que uma das quatro condições necessárias nunca ocorram.
- ➡ Para evitar a condição da exclusão mútua:
 - Nenhum dos recursos é dedicado.
 - Como alguns recursos devem ser dedicados, deveremos definir um outro critério para garantir o uso exclusivo destes recursos.

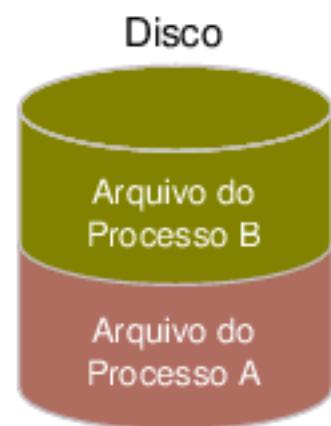


Se um outro processo, o B, decidir também criar um arquivo para gerar a sua saída, antes do processo A fechar o seu arquivo, o disco poderá ficar cheio, e com isso, os processos ficarão bloqueados, pois não serão capazes de gravar as suas saídas no disco.



Prevenção de impasses

- ➡ A idéia é evitar a ocorrência dos impasses garantindo que uma das quatro condições necessárias nunca ocorram.
- ➡ Para evitar a condição da exclusão mútua:
 - Nenhum dos recursos é dedicado.
 - Como alguns recursos devem ser dedicados, deveremos definir um outro critério para garantir o uso exclusivo destes recursos.



Se o processo spooler não começar a imprimir um arquivo antes de este ser fechado, este irá também bloquear ao tentar imprimir o arquivo do processo A, ou do processo B. Logo, teremos um impasse, mas agora o recurso envolvido será o espaço no disco, e não mais a impressora.



Prevenção de impasses



Para evitar a condição de segura e espera:

- O processo deve solicitar todos os recursos, e começar a sua execução somente após de obter estes recursos, mas:
- Nem sempre é possível saber todos os recursos necessários.
- Não conseguiremos otimizar o uso dos recursos:



Se um processo conseguir obter a impressora e o CDROM, e, antes de imprimir, lê algumas informações do CDROM, e leva várias horas para processá-las, a impressora ficará ociosa durante estas horas.

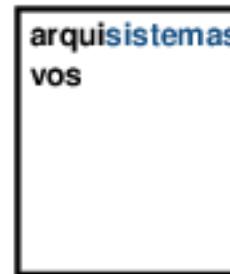
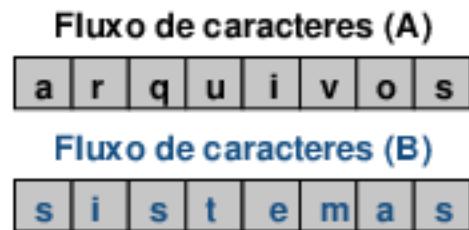
- O processo também poderia, ao solicitar um recurso:
 - Liberar todos os recursos que possui.
 - Se a solicitação puder ser atendida, o processo recebe de volta os recursos que possuia, e o recurso que solicitou.
 - Caso contrário, o processo perde os recursos que possuia, e espera até conseguir obter todos os recursos necessários.

Prevenção de impasses



Para evitar a condição de nenhuma preempção:

- O sistema poderá tirar um recurso do processo, se necessário.
- O problema é que alguns recursos devem ser não-preemptivos, para evitar erros de computação.



Esta idéia claramente não é boa, pois vimos anteriormente que, no caso da impressora, as saídas dos processos seriam intercaladas.



A condição de espera circular pode ser evitada se um processo somente puder usar um recurso em um dado intervalo de tempo:



Suponha que um processo deseja imprimir um arquivo bem grande da unidade de fita. Se o processo não puder usar estes dispositivos ao mesmo tempo, além de a impressão ser incorreta, perderemos um bom tempo para reposicionar a fita na próxima posição do arquivo a ser lido.

Prevenção de impasses

- Um outro modo de evitar a condição de espera circular é o de numerar os recursos do sistema:
- Os processos devem solicitar os recursos em ordem crescente.
 - Neste caso, o grafo de recursos não poderá ter ciclos.
 - Uma alternativa menos restritiva seria a seguinte:
 - O processo somente pode solicitar um recurso cujo número é maior do que os números de todos os seus recursos.

Recursos numerados:

1. CDROM
2. Impressora
3. Plotadora
4. Unidade de fita
5. Braço autômato

A

i

B

j

Os recursos serão numerados em ordem crescente. Nesta idéia, uma solicitação de um processo somente será aceita se o número do recurso for maior do que o de todos os recursos que este processo possui.

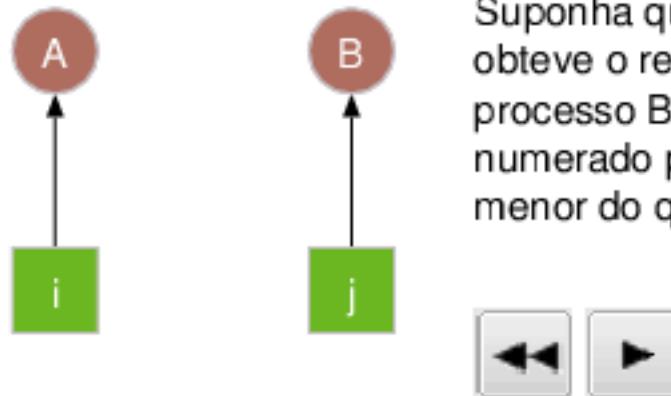


Prevenção de impasses

- Um outro modo de evitar a condição de espera circular é o de numerar os recursos do sistema:
 - Os processos devem solicitar os recursos em ordem crescente.
 - Neste caso, o grafo de recursos não poderá ter ciclos.
 - Uma alternativa menos restritiva seria a seguinte:
 - O processo somente pode solicitar um recurso cujo número é maior do que os números de todos os seus recursos.

Recursos numerados:

1. CDROM
2. Impressora
3. Plotadora
4. Unidade de fita
5. Braço autômato



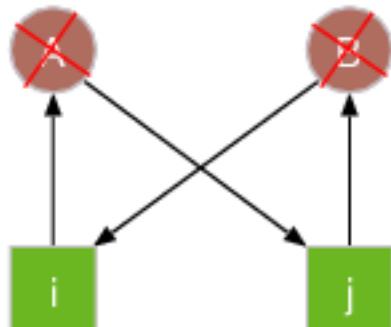
Suponha que o processo A solicitou e obteve o recurso numerado por i, e que o processo B solicitou e obteve o recurso numerado por j. Suponha ainda que i é menor do que j.

Prevenção de impasses

- Um outro modo de evitar a condição de espera circular é o de numerar os recursos do sistema:
 - Os processos devem solicitar os recursos em ordem crescente.
 - Neste caso, o grafo de recursos não poderá ter ciclos.
 - Uma alternativa menos restritiva seria a seguinte:
 - O processo somente pode solicitar um recurso cujo número é maior do que os números de todos os seus recursos.

Recursos numerados:

1. CDROM
2. Impressora
3. Plotadora
4. Unidade de fita
5. Braço autômato



Se os processos puderem solicitar os recursos sem restrição, e se A solicitar j antes de B o liberá-lo, e B solicitar i antes de A o liberá-lo, teremos um impasse.

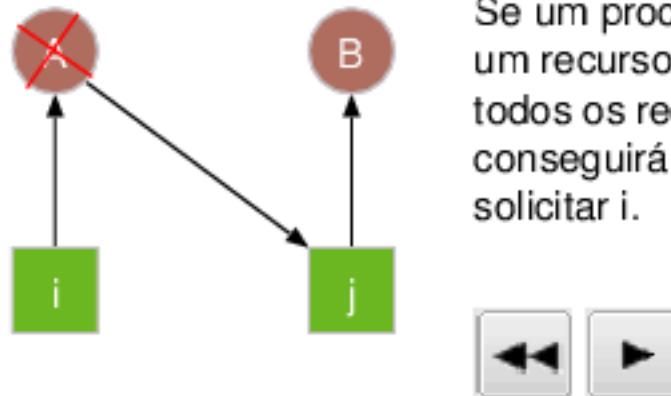


Prevenção de impasses

- Um outro modo de evitar a condição de espera circular é o de numerar os recursos do sistema:
 - Os processos devem solicitar os recursos em ordem crescente.
 - Neste caso, o grafo de recursos não poderá ter ciclos.
 - Uma alternativa menos restritiva seria a seguinte:
 - O processo somente pode solicitar um recurso cujo número é maior do que os números de todos os seus recursos.

Recursos numerados:

1. CDROM
2. Impressora
3. Plotadora
4. Unidade de fita
5. Braço autômato



Se um processo puder solicitar somente um recurso cujo número é maior do que todos os recursos que possui, então A conseguirá solicitar j, mas B não poderá solicitar i.



Prevenção de impasses

- Um outro modo de evitar a condição de espera circular é o de numerar os recursos do sistema:
- Os processos devem solicitar os recursos em ordem crescente.
 - Neste caso, o grafo de recursos não poderá ter ciclos.
 - Uma alternativa menos restritiva seria a seguinte:
 - O processo somente pode solicitar um recurso cujo número é maior do que os números de todos os seus recursos.

Recursos numerados:

1. CDROM
2. Impressora
3. Plotadora
4. Unidade de fita
5. Braço autômato

A

i

B

j

Depois de B liberar o recurso j, A poderá obter este recurso, e terminar. B somente poderá solicitar o recurso i após liberar o recurso j, e todos os outros recursos cujo número é maior do que i.

