Sistemas Operacionais

Aula 09 - Gerenciamento de Processos

Apresentação

Como você já viu nas aulas anteriores, quando usamos um computador podemos executar vários programas ao mesmo tempo: conseguimos mandar um e-mail enquanto escutamos uma música, ou mesmo entrar no Facebook enquanto estamos digitando um texto usando um editor de texto aberto.

Nesta aula, vamos investigar como os processos (programas em execução) compartilham os recursos do nosso computador (processador, impressora, arquivos etc.) e veremos também quais os principais problemas que surgem quando processos compartilham recursos e algumas alternativas para solucioná-los.

Objetivos

- Distinguir os conceitos de paralelismo e pseudoparalelismo no contexto da programação concorrente.
- Entender o ciclo de vida de um processo.
- Descrever as principais dificuldades associadas ao gerenciamento de processos.
- Definir o conceito de *deadlock* e condição de corrida.

Pseudoparelelismo: Mais de um Processo Compartilhando um Processador

Vídeo 01 - Pseudoparelelismo

Como você já percebeu, um computador pode executar diversas atividades ao mesmo tempo. Você pode digitar um texto enquanto escuta sua música preferida e gravar na nuvem as fotos de suas últimas férias. Imagino que você deve estar se perguntando: Como isso é possível, se muitos computadores possuem apenas um processador? Além disso, tem outra coisa: mesmo os computadores que possuem dois ou mais núcleos de processamento podem executar para nós uma quantidade de programas bem superior à quantidade de processadores. Tudo isso pode parecer estranho, mas vamos procurar entender como isso é possível no decorrer desta aula.

Hoje em dia, são comuns os computadores com mais de um processador; você pode, por exemplo, identificá-los através de termos técnicos que indicam a quantidade de núcleos de processamento. Verifique nos anúncios e propagandas de computadores termos como Dual Core, Quad Core, os quais indicam respectivamente dois e quatro núcleos de processamento.

Digamos que temos um simples computador com apenas um processador. Como o sistema operacional faz com que possamos, por exemplo, escutar música e navegar na internet ao mesmo tempo? Vale a pena lembrar que o próprio sistema operacional é também um programa e, portanto, ele também precisa do processador para executar. A solução encontrada para essa questão é: compartilhar! Os processos precisam compartilhar o processador.

Você já pode ter vivido o seguinte cenário: existe apenas um único jogo eletrônico e você e todos seus amigos querem jogar. Para resolver o impasse, você faz o seguinte: “Cada um usa o *game* por 30 minutos e depois passa para o próximo da fila de interessados em jogar!”. Pois bem, podemos ver os processos como um conjunto de amigos que querem usar um recurso que é limitado (o processador); assim, o que o sistema operacional precisa fazer, na condição de gerente, é disponibilizar o processador para cada processo por um determinado período de tempo.

Dessa forma, temos a impressão de que os programas estão sendo executados ao mesmo tempo (em paralelo). Mas o que acontece na verdade é que os processos formam uma fila e cada um usa o processador por um período de tempo pequeno (na ordem de milésimos de segundo) e passa para o processo seguinte na fila. E assim continua até que todos os processos usem o processador.

Dando continuidade, a CPU é passada novamente para o processo no início da fila e tudo se repete. Assim, teremos a impressão de que estamos com diversos aplicativos em aberto sendo executados simultaneamente. Essa impressão é conhecida como **pseudoparalelismo**, pois apesar dos programas não estarem realmente sendo executados ao mesmo tempo (em paralelo), eles ficam se alternando numa frequência que aparenta uma operação simultânea de diversos aplicativos.

Você já deve ter visto um show de malabarismo com pratos que giram em cima de bastões. Como cada prato precisa por um momento do malabarista para dar novo impulso no bastão e continuar girando, o malabarista procura distribuir o seu tempo entre os pratos para que todos continuem girando durante toda a apresentação. Então, considerando os pratos como se fossem processos e o malabarista com o processador, podemos entender melhor o pseudoparalelismo existente

Fica então sob a responsabilidade do sistema operacional gerenciar os processos, determinando a ordem de uso do processador, de forma a garantir que cada um tenha acesso e tempo suficiente para executar normalmente.

Gerenciando Processos

Vídeo 02 - Operações Típicas para Gerenciamento de Processos

Para desempenhar o papel de gerente de processos, o sistema operacional precisa ter capacidades/operações para:

1. criar processos;
2. reservar memória;
3. colocar os processos numa fila de espera para uso do processador.

Deve-se lembrar que o próprio sistema operacional é formado por um conjunto de vários processos que também compartilham a CPU para serem executados. A Figura 1 representa o compartilhamento da CPU entre os diferentes processos. Nessa figura, são exemplificadas diferentes situações de alocação da CPU, as quais são identificadas na figura por números. Uma breve descrição desses exemplos é apresentada a seguir.

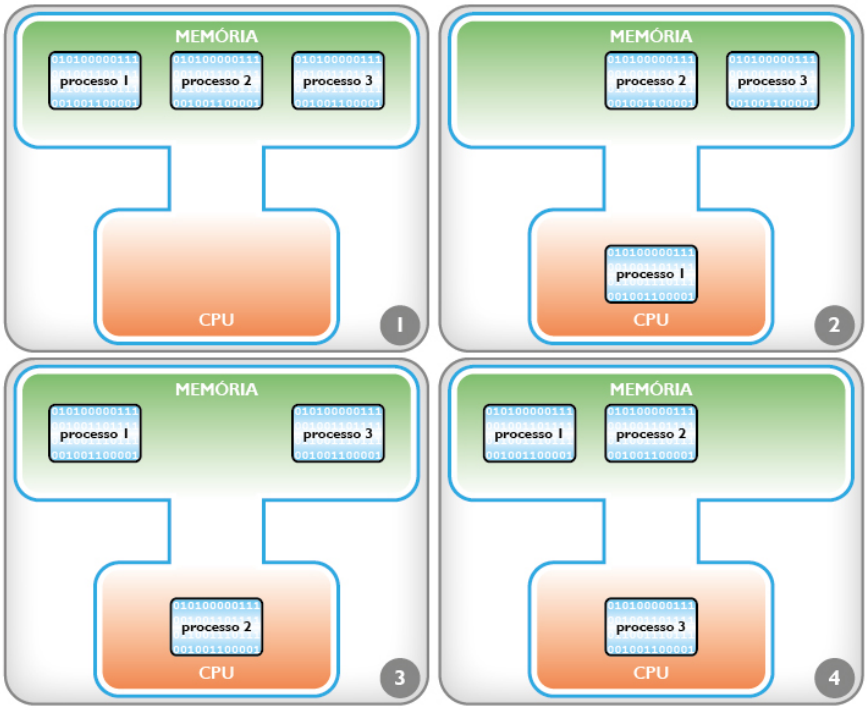

Figura 1 - Processos compartilhando a CPU
Fonte: autoria própria.

Os sistemas operacionais permitem que processos (através de chamadas de sistema) criem outros processos. Pense nisso como uma árvore genealógica de uma família que pode aumentar a quantidade de membros à medida que o tempo passa. Portanto, os processos podem ser "pai" e "filho" de outros processos. A parte superior da Figura 1, lado esquerdo (situação 1), apresenta um exemplo do sistema com 3 processos em memória para serem executados (processos 1, 2 e 3) e, do lado direito (situação 2), um exemplo em que o processo 1 foi escolhido para executar. Nesse caso, os processos 2 e 3 continuam em memória, aguardando serem escolhidos para executarem. As situações 3 e 4, parte inferior da figura, ilustram a escala dos processos 2 e 3, respectivamente.

Por exemplo, quando você executa o programa Internet Explorer para navegar na internet a partir de um computador com sistema operacional Windows, na verdade você está solicitando a criação do iexplorer.exe (processo associado ao programa Internet Explorer) ao explorer.exe (processo genérico do sistema responsável por fornecer comandos básicos e gerenciar o ambiente gráfico padrão do Windows). A Figura 2 mostra a tela de um computador que está executando o Process Explorer, um programa de gerenciamento de processos cuja tela inicial mostra todos os processos que estão executando em seu computador e a relação

entre eles (por exemplo, se um processo é pai de outro). Observe como o processo iexplorer.exe está situado entre os processos filhos do explorer.exe.

Figura 2 - Relação pai x filho entre processos
Fonte: autoria própria.

Caso você queira visualizar apenas quais processos estão sendo executados, o Windows possui o gerenciador de tarefas do sistema. Para acessá-lo, basta pressionar os botões Ctrl, Alt e Del simultaneamente e escolher a opção “Iniciar Gerenciador de Tarefas”, aba “Processos” (Figura 3). Assim, você terá acesso a essa importante ferramenta do sistema e poderá utilizá-la não só para visualizar os processos, mas também, caso ocorra algum problema com um processo específico, poderá encerrá-lo.

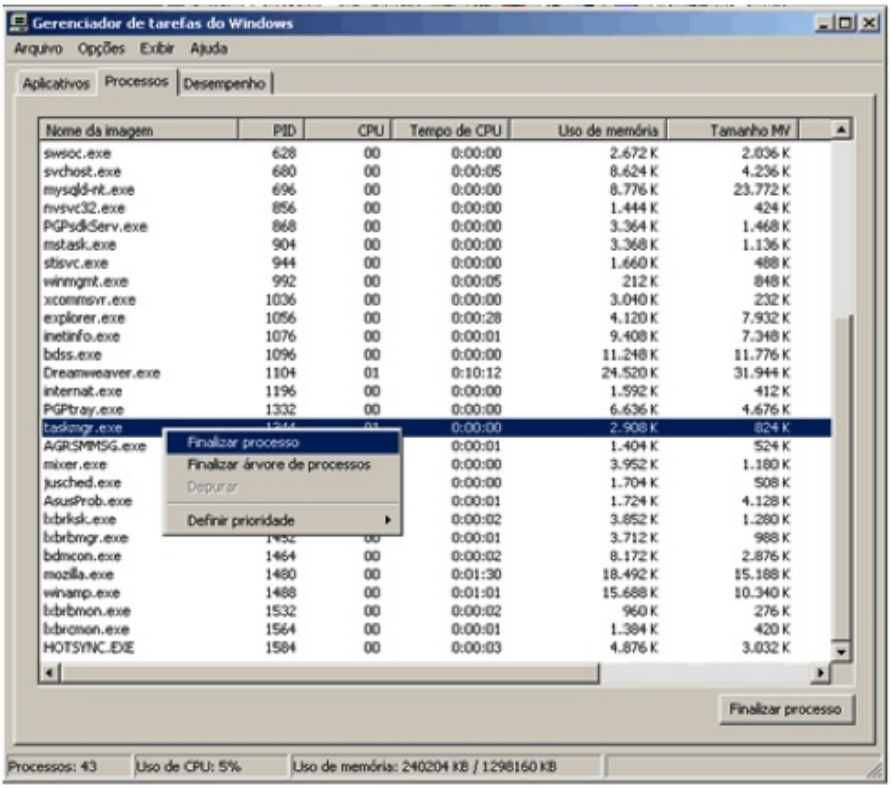

Figura 3 - Gerenciador de tarefas do Windows
Fonte: http://www.linhadefensiva.org/2004/10/gerenciador-de-tarefas/ Acesso em: 08 mai. 2015

Sugestão de Programas Úteis

Para visualizar a árvore de processos no Windows, existe o programa Process Explorer (Figura 4), que traz muito mais informações que o gerenciador de tarefas do Windows. Além de apresentar a estrutura de processos, indicando o parentesco entre eles, apresenta dados referentes a cada processo, como a quantidade de memória utilizada por cada um. Vale a pena conferir, ele é um *software* gratuito e está disponível no seguinte *link* para download: <http://www.baixaki.com.br/download/process-explorer.htm>.

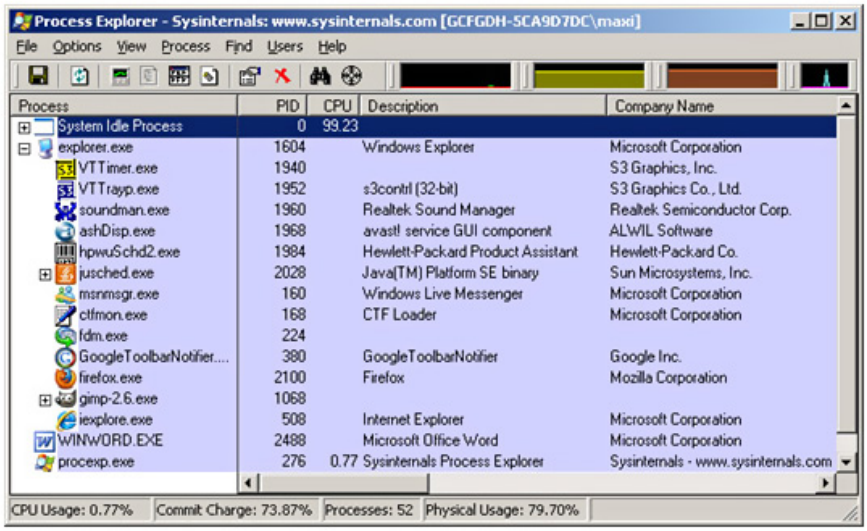

Figura 4 - Tela do Process Explorer

Nesta aula, vamos investigar como os processos (programas em execução) compartilham os recursos do nosso computador (processador, impressora, arquivos etc.) e veremos também quais os principais problemas que surgem quando processos compartilham recursos e algumas alternativas para solucioná-los.

Quando o sistema operacional é inicializado, ou seja, quando o computador é ligado, o primeiro processo a ser executado é o processo que representa o sistema operacional. Os demais processos são criados a partir dele através de chamadas de sistema. Por exemplo, nos sistemas operacionais Windows (NT, 2000 ou XP), o processo inicial se chama *smss.exe* e a chamada de sistema responsável por criar os demais processo se chama *CreateProcess()*. Já no Sistema Operacional Linux, o processo que representa o sistema operacional se chama *init* e a chamada de sistema responsável por criar os demais processos a partir dele se chama *fork()*. A Figura 5 mostra uma árvore de processos que está sendo executada em um computador com Sistema Operacional Linux (Para visualizar essa árvore de processos, o comando utilizado no Linux foi o comando pstree.).

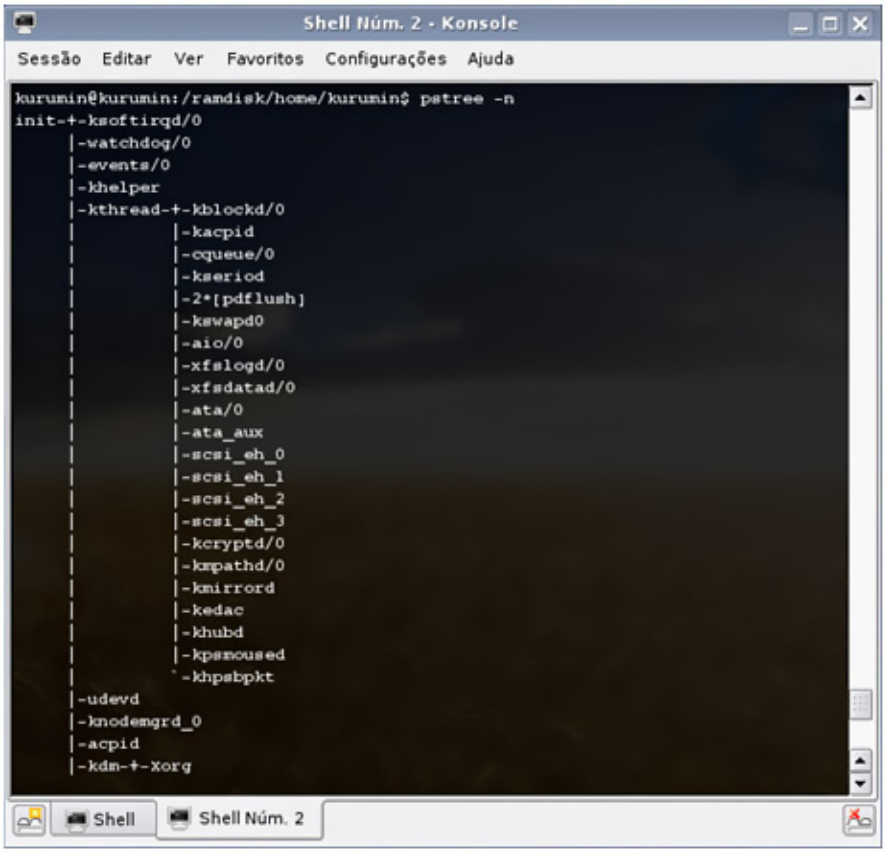

Figura 5 - Árvore de processos no Linux

Os vários processos criados competem entre si pela atenção do processador a cada instante, cabendo ao gerenciador de processos do sistema operacional estabelecer uma ordem entre eles.

Em diferentes momentos, um processo pode estar utilizando-se do processador, ou simplesmente aguardando algum evento (por exemplo, a digitação de uma informação solicitada). Se ocorrer uma situação em que um processo, mesmo tendo o processador disponível, esteja impossibilitado de prosseguir a execução, o sistema operacional deve bloqueá-lo para evitar desperdício no uso do processador, deixando os demais processos disputarem pelo uso do processador. Dessa forma, de uma maneira simplificada, podemos dizer que os processos podem estar em três estados, que seriam:

1. **executando:** utilizando o processador;
2. **pronto:** o processo está em condição de ser executado, porém, encontra-se aguardando a disponibilização de um processador;
3. **bloqueado:** o processo está impossibilitado de usar o processador até que algum evento externo aconteça.

Vídeo 03 - Estados de um Processo

As possíveis transições entre os estados de um processo estão representadas na Figura 6.

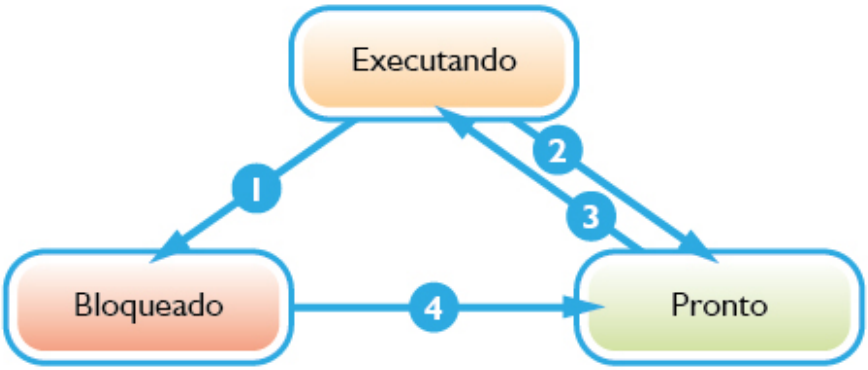

Figura 6 - Estados de um processo

A transição 1 indica o momento em que um processo se depara com uma situação em que não pode continuar a execução (normalmente devido à espera de uma entrada de dados). Essa situação é reconhecida pelo sistema operacional através de um mecanismo denominado sistema, cujo efeito é colocar o processo no estado bloqueado.

Na transição 4, o processo é desbloqueado por um evento externo (normalmente acontece quando o dado que estava sendo aguardado chega ao processo); basicamente, um processo que está monitorando as entradas de diversos outros processos fica responsável em desbloquear.

A transição 2 indica que o processo já passou tempo demais na CPU e deve dar lugar a outro processo, enquanto que a transição 3 indica que um processo pronto pode ser escolhido para retornar à CPU. As transições 2 e 3 dependem de um agendador de processos, que é um módulo do sistema operacional responsável por gerenciar o tempo de uso do processador. Esse agendador é normalmente citado nos livros de sistemas operacionais com o nome de escalonador.

Observe que enquanto os processos têm controle sobre as transições 1 e 4, o sistema operacional determina quando as transições 2 e 3 ocorrem e, assim, os processos não tomam nem conhecimento quando elas ocorrem.

Agora que você conhece os estados possíveis de um processo, veremos como realizar as transições entre esses estados e a importância de manter os dados relacionados aos processos salvos.

Atividade 01

1. Apresente as principais características dos gerenciadores de processos do Windows e do Linux. Quais as diferenças entre eles?

Ciclo de Vida de um Processo

Vídeo 04 - Gerência de Processos

Assim como o ser humano tem um ciclo de vida – o homem nasce, cresce, reproduz e morre – os processos também têm seu ciclo de vida. Para gerenciar os processos e seus estados, o sistema operacional mantém guardados dados sobre todos os processos presentes na memória. Dessa forma, o agendador de processos pode obter informações úteis na hora de decidir qual processo terá acesso ao

processador. Lembre-se do exemplo dos amigos que estão jogando videogame; é necessário guardar os nomes e horários de cada jogador em uma tabela de forma a distribuir o *game* de forma justa.

Vídeo 05 - Escalonamento por Antiguidade

Vídeo 06 - Escalonamento por Rodízio

De forma análoga, o sistema operacional usa uma tabela de processos, na qual mantém dados, como o estado atual de cada processo, um marcador que indica qual a próxima instrução deve ser executada e tudo mais que for relevante para o gerenciamento do processo.

Mas por que guardar tanta informação sobre os processos? Não seriam suficientes os dados presentes na área de memória reservada para cada processo e uma tabela que indicasse o endereçamento de cada processo memória?

Para entender melhor a importância da tabela de processos e do agendador de processos, vamos comparar o agendador com o trabalho de um continuísta (Profissional responsável por manter, durante as diversas cenas e montages de produções televisivas e cinematrográficas, a harmonia do enredo, falas, sonoplastia e imagens.) num set de filmagens onde várias cenas serão rodadas.

Num filme, devido a questões de agendas e compromissos dos atores e profissionais envolvidos, as cenas podem ser filmadas em diferentes ordens e em dias diferentes. Para que a cena possa prosseguir normalmente, é papel do continuísta saber dos detalhes entre as filmagens de uma cena, evitando as falhas de continuidade quando o filme estiver pronto. Já imaginou como seria estranho se surgissem

objetos do nada no meio de uma cena! A **Figura 7** mostra um erro de continuidade em uma cena do filme “Amor a Segunda Vista” (de título original em inglês *Two Weeks Notice*).

Figura 7 - Cena de um filme com erros de continuidade
Fonte: <http://www.moviemistakes.com/picture38273> Acesso em: 7 out. 2011.

De forma análoga ao trabalho do continuísta, é também responsabilidade do sistema operacional, através do agendador de processos, manter as informações dos processos consistentes com os respectivos estados (na tabela de processos), de modo que, quando eles forem se alternando no uso do processador, não ocorra descontinuidade.

Deve-se destacar que a função do sistema operacional não se limita apenas ao agendamento dos processos. Em muitos casos, apesar de os processos competirem entre si pelo uso do processador e de cada um ter o seu próprio espaço de memória protegido pelo sistema operacional,

existe a necessidade de estabelecer cooperações entre processos quando se deseja executar tarefas muito complexas, de forma a evitar interferências de outros processos.

Nesse caso, a colaboração e a comunicação entre eles, seja para troca de informação ou simplesmente por uma sequência de ações dos processos cooperantes, precisam ser executadas com um certo cuidado, sob pena de gerar situações inconsistentes. Essa cooperação entre os processos deve também ser coordenada pelo sistema operacional.

Comunicação Interprocessos

O ser humano não vive isoladamente; precisamos a todo o momento nos comunicar uns com os outros, seja verbalmente, seja por escrito ou através de gestos (por exemplo, através de um simples sinal de mão que você faz para seu colega numa partida de futebol dizendo que você está livre para receber a bola). No mundo da computação, mais especificamente quando falamos de processos, a comunicação é também muito importante. Por exemplo, quando mais de um processo compartilha o acesso da impressora, os processos precisam saber quando a impressora está disponível para ser utilizada.

Vídeo 07 - Interprocessos

Gerenciar a comunicação entre processos é uma tarefa um tanto complicada. Para entender melhor essa complexidade, imaginemos a seguinte situação: vamos supor que você e seu amigo desejam comprar o mesmo livro em um mesmo site de vendas e ambos realizam o pedido quase ao mesmo tempo. Após o pedido, o livro deverá ser colocado no seu carrinho de compra.

De acordo com a Figura 8, vemos que no estoque da loja virtual existe apenas uma unidade do livro que você deseja comprar (1). O processo da primeira compra verifica que o estoque do produto é igual a 1 e portanto

ele registra essa informação entre seus dados internos (2); antes que esse processo adicione o livro no carrinho virtual de compras, o segundo processo verifica que ainda existe uma unidade do livro e registra essa informação internamente (3). Logo em seguida, ambos os processos gravam estoque zero para o sistema (4 e 5). E, no final, ambos os carrinhos virtuais estão com o mesmo livro de posse (6), provocando um erro.

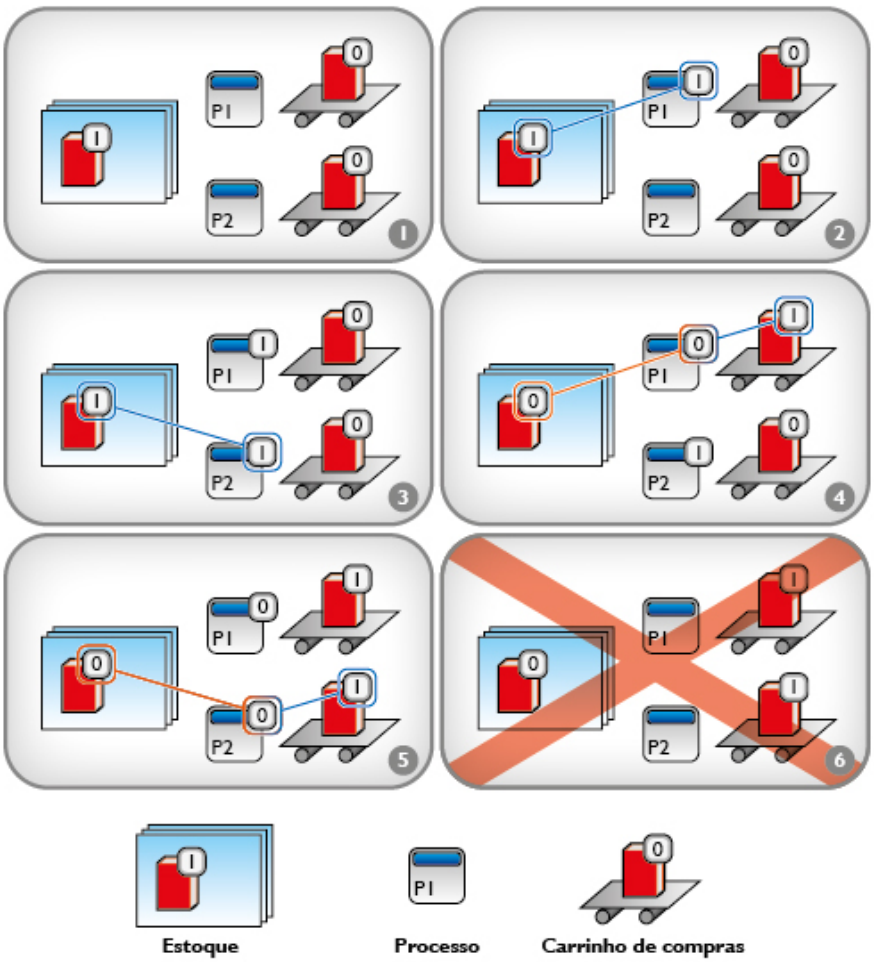

Figura 8 - Falha de comunicação na compra de um livro

Uma solução simples para evitar o erro seria deixar o primeiro processo gravar o estoque com zero antes que fosse verificada a situação do estoque pelo segundo processo; entretanto, no mundo real, a ordem de execução das ações de cada processo é imprevisível.

Essa situação é conhecida como condição de corrida, pois a velocidade de execução dos processos que competem pelo mesmo recurso (no exemplo, o livro) pode afetar o resultado final, colocando-o numa situação inconsistente.

Para solucionar o problema, é preciso encontrar uma maneira de proibir que mais de um processo leia e grave dados compartilhados ao mesmo tempo. Essa abordagem, utilizada para solucionar os problemas das disputas nos acessos a recursos comuns, é chamada nos livros como mecanismo de exclusão mútua, ou seja, uma forma de garantir que se um processo está utilizando-se de um recurso compartilhado, os outros processos estão impedidos de acessar o mesmo recurso.

A parte do programa que eventualmente pode levar a uma condição de corrida é conhecida como **região crítica**. A ação de alterar o estoque no exemplo anterior pode ser considerada então como uma região crítica e, caso ela esteja inserida em um programa, deve ser tratada de forma a garantir que toda vez que um processo começar a executá-la, ele realizará a ação completa, com exclusão mútua. Assim, considerando o exemplo anterior, o processo precisa inicialmente obter a autorização para o acesso exclusivo ao estoque e, na sequência, executar toda a ação associada, para então, ao final, disponibilizar o estoque atualizado para os demais processos.

É, portanto, papel do sistema operacional garantir que os processos cooperem de forma correta e eficientemente. Para tanto, o sistema operacional trabalha para assegurar que algumas condições de execução sejam atendidas, entre elas podemos citar:

1. não pode existir mais de um processo dentro de uma mesma região crítica;
2. nenhum processo deve esperar eternamente para entrar em sua região crítica.

Garantidas essas e outras condições, o sistema operacional permitirá que os processos se utilizem de recursos compartilhados sem a ocorrência de condições de corrida e sem situações de impasse na obtenção de exclusividade no acesso aos recursos.

Atividade 02

1. Na internet, existem diversos exemplos que podem gerar condições de corrida. Cite um deles, explicando quais ações devem pertencer à região crítica e, em consequência, devem ser executadas com exclusão mútua.

Deadlock: Impasse entre Processos

Vídeo 08 - Deadlock

Quando dois ou mais processos ficam impedidos de continuar suas execuções, ou seja, quando estiverem bloqueados aguardando a liberação de recursos que estão alocados entre eles, dizemos que essa situação é denominada como *deadlock*. É difícil identificar a ocorrência de *deadlock*s; basicamente essa situação de impasse pode ser representada por um grafo, conforme exemplo apresentado na Figura 9.

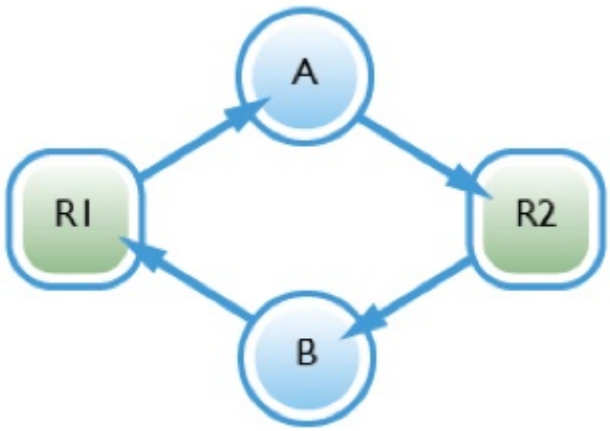

Figura 9 - Processos A e B em *deadlock* devido aos recursos R1 e R2
Fonte: autoria própria.

No caso da Figura 9, o processo “A” está usando os recursos de R1 e precisa, para concluir sua tarefa, usar os recursos de R2; por sua vez, o processo “B” está usando os recursos de R2, porém, precisa dos recursos de R1 para concluir sua tarefa. Deve-se destacar que os recursos R1 e R2 são críticos e, portanto, devem ser usados com exclusão mútua. Esse estado de colapso corresponde ao que chamamos de *deadlock*.

Outro exemplo envolvendo o conceito de *deadlock* é apresentado na Figura 10. No exemplo, supomos que quatro trens estão chegando a uma encruzilhada ao mesmo tempo. Não há sinal e assume-se que nenhum trem pode dar marcha a ré. Como não existe nenhum meio de controle para acessar a encruzilhada, os trens ficarão bloqueados, ou seja, sem poder avançar por tempo indeterminado até que algum seja rebocado. Tal situação também representa um estado de *deadlock*, em que fazemos uma analogia dos trens com processos e a encruzilhada como uma região crítica.

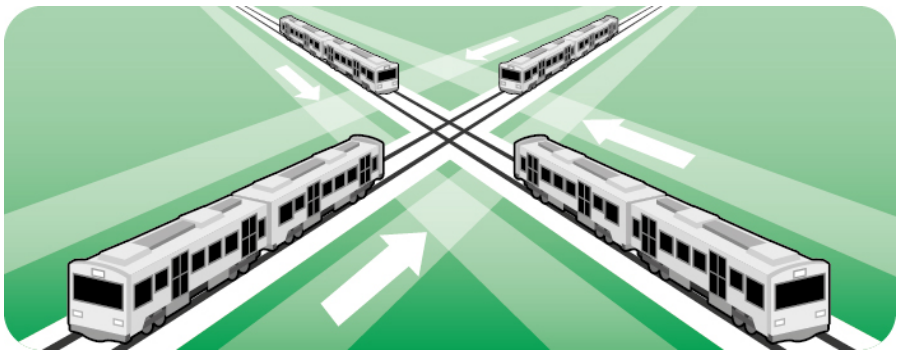

Figura 10 - Quatro trens chegam ao mesmo tempo em uma encruzilhada sem sinal
Fonte: autoria própria.

Observe que, quanto mais recursos compartilhados forem necessários para execução de processos, há mais riscos desses processos atingirem uma situação de impasse com outros processos. Um exemplo simples seria um sistema envolvendo processos para manipulação de banco de dados, cujas diferentes tabelas podem ser consideradas como recursos compartilhados. Veja um exemplo no qual pode ocorrer um *deadlock*.

Para efetuar uma venda, um processo é criado no computador onde estão as informações no banco de dados. O sistema precisa alocar as tabelas de estoque dos produtos e de clientes. Como vários vendedores podem registrar vendas ao mesmo tempo, dois processos podem receber a autorização de acesso a uma das tabelas em questão enquanto ficam aguardando a liberação da outra; assim, tanto os processos ficam bloqueados, como os recursos ficam indisponíveis até que essa situação de impasse seja resolvida. E, afinal de contas, quais são as estratégias para tratamento de *deadlock*s?

Deadlocks: Métodos de Tratamento

Existem três estratégias básicas para o tratamento de *deadlock*s, que serão discutidas a seguir.

1. Definir métodos de comunicação interprocessos que evitem o surgimento de *deadlock*s.
2. Detectar situações de *deadlock* e definir uma forma de restaurar o sistema, resolvendo o impasse.
3. Ignorar o problema e torcer que ele não ocorra. Conhecido como algoritmo do avestruz.

Por incrível que pareça, a estratégia mais usada nos sistemas operacionais é a de ignorar os problemas de *deadlock*, pois os possíveis tratamentos desse tipo de situação podem acarretar em perdas de desempenho do sistema, comprometendo a execução dos processos. Assim, é mais comum que os próprios softwares procurem evitar situações de *deadlock* na lógica em seus códigos e, no caso de ocorrer alguma situação de impasse, é preferível encerrar a execução de um processo ou até reiniciar o computador em casos mais extremos.

Atividade 03

1. Diversos exemplos na literatura de sistemas operacionais procuram demonstrar situações de *deadlock*. São exemplos simples, como o clássico exemplo dos cinco filósofos jantando. Procure na internet dados sobre esse exemplo e apresente como pode ocorrer um *deadlock*.

Término de Processos

Para concluir o gerenciamento de processos, existem nos sistemas operacionais chamadas de sistema que possibilitam que o processo informe o fim de sua execução antes mesmo do fim do seu código. Uma chamada comum, presente em muitos sistemas operacionais que faz exatamente isso é a **exit()**.

Além dessa função, os sistemas operacionais também disponibilizam funções que permitem encerrar processos individualmente, a chamada que faz isso é a **kill()**.

Leitura Complementar

Uma boa descrição sobre gerenciamento de processos pode ser obtida no link <http://www.das.ufsc.br/~romulo/artigos/Romulo-Carissimi-Simao-Rita2001.pdf>.

Nele você encontra o artigo:

OLIVEIRA, R. S.; CARISSIMI, A. S.; TOSCANI, S. S. Sistemas operacionais. **Revista de Informática Teórica e Aplicada – RITA**, v. 8, n. 3, dez. 2001.

E que posteriormente serviu de base para a publicação de um livro de mesmo nome.

Resumo

Nesta aula, você viu que o gerenciamento de processos é um dos módulos fundamentais dos sistemas operacionais e é essencial o entendimento de seus principais conceitos. Foram apresentadas as ideias de processo, pseudoparalelismo e alternância entre processos, comunicação interprocessos e problemas relacionados, tais como condições de corrida e *deadlock*s.

Autoavaliação

1. O que significa dizer que na gerência de processos acontece o que chamamos de pseudoparalelismo?
2. Qual o ciclo de vida de um processo?
3. Por que dizemos que o gerenciamento de processos é uma atividade complexa?
4. Defina, com suas palavras, o que é condição de corrida. Como esse conceito está relacionado com o conceito de área protegida?
5. O que é o *deadlock*?

Referências

PROCESSO Windows. Disponível em <http://pt.kioskea.net/contents/processus/>. Acesso em: 7 out. 2010.

SISO 2.0: simulador de sistema operacional: módulo de *deadlock*. Disponível em <http://www.martins.eti.br/2009/03/siso-20-simulador-de-sistema.html>. Acesso em: 7 out. 2011.

TANENBAUM, Andrew S.; WOODHULL, Albert S. **Sistemas operacionais:** projeto e implementação. Trad. Edson Furmankiewicz. 2. ed. Porto Alegre: Bookman, 2000.