Arquitetura de Computadores

Gerenciamento de processos

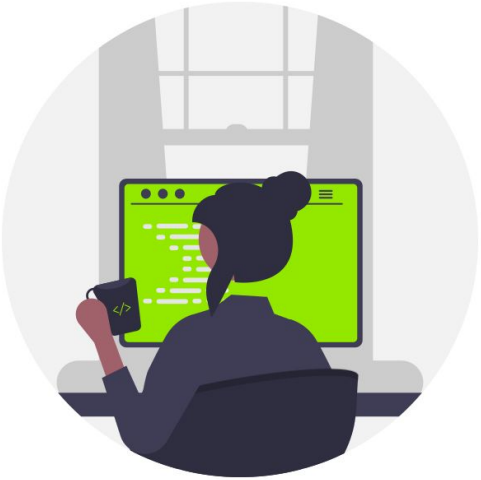
Quem sou eu

Júlio César Andrade

Bacharel em Engenharia de Computação - UEFS
Especialista em User Experience - UNIFACS
Mestrando em Ciências da Computação - UEFS

O que é um processo?

- Um processo é basicamente um programa em execução (TANENBAUM, 2014)
- Um processo é o contêiner que detém todas as informações necessárias para rodar um programa (TANENBAUM, 2014)



SO e processos

O sistema operacional é responsável pelas seguintes atividades em relação à gerência de processos:

a) Criar e excluir processos de usuário e de sistema;

b) Suspender e retomar processos;

c) Fornecer mecanismos para a sincronização de processos;

d) Fornecer mecanismos para a comunicação de processos;

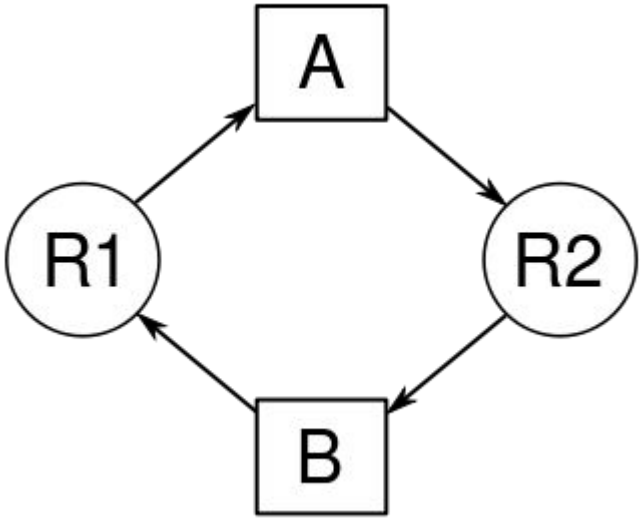
e) Fornecer mecanismos para o tratamento de **deadlocks**

Mas o que são deadlocks?

Deadlock

Deadlock caracteriza uma situação em que ocorre um impasse, onde dois ou mais processos ficam impedidos de continuar suas execuções, ou seja, ficam bloqueados.

Exemplo: Um processo aguarda a liberação de um recurso que está sendo utilizado por outro processo, que, por sua vez, aguarda a liberação de outro recurso alocado ou dependente do primeiro processo.



Processo vs programa

Programa: entidade estática e permanente

- composto por uma seqüência de instruções: passivo sob o ponto de vista do sistema operacional

Processo: entidade dinâmica

- altera seu estado a medida que avança sua execução;
- o processo é uma abstração que representa um programa em execução;

Componentes de um processo

Programa Executável: É o código do programa que está armazenado no disco. Este é o arquivo binário que contém as instruções a serem executadas.

Dados do Processo: Isso inclui variáveis, buffers e outras informações necessárias para a execução do programa. Os dados do processo são geralmente armazenados na memória durante a execução.

Componentes de um processo

Contexto do Processo: Inclui o conteúdo dos registradores do processador, o contador de programa e outras informações importantes do estado do processador. Isso permite que o sistema operacional salve e restaure o estado do processo quando necessário.

Espaço de Endereçamento: Cada processo tem seu próprio espaço de endereçamento, que contém instruções e dados. Isso fornece isolamento entre processos, impedindo que um processo acesse diretamente a memória de outro.

Analogia entre um Processo e um Cozinheiro

Imagine um engenheiro com dotes culinários fazendo um bolo:

- Receita = programa
- Engenheiro cozinheiro = processador (CPU)
- Ingredientes = dados de entrada.

Processo é a atividade desempenhada pelo cozinheiro em ler a receita, buscar os ingredientes e assar o bolo.

Características/Propriedades de um processo

- Um processo tem execução sequencial;
- O resultado da execução de um processo independe da velocidade do processador em que for executado;
- O mesmo programa executado por dois usuários gera dois processos.

Características/Propriedades de um processo

Um programa pode gerar (criar) vários processos.

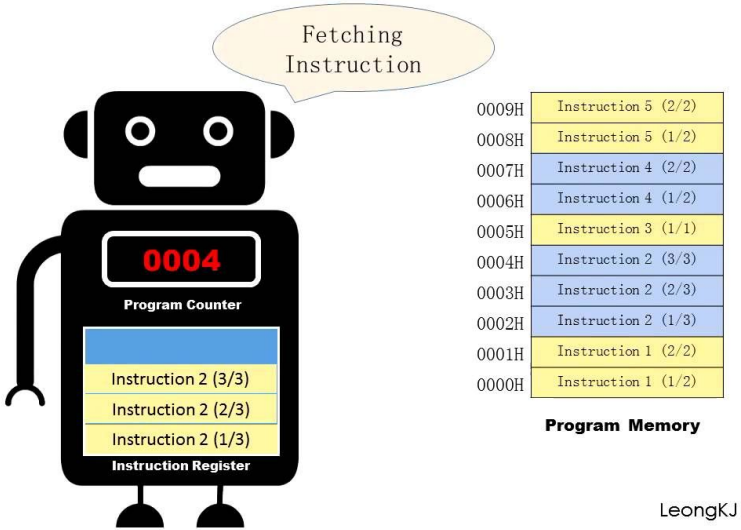
Um processo tem duas partes:

- Ativa - fluxo de controle
- Passiva - espaço de endereçamento (memória, registradores, arquivos)

Contador de programa

O contador de programa é um registrador especial em um processador que indica a posição da próxima instrução a ser executada.

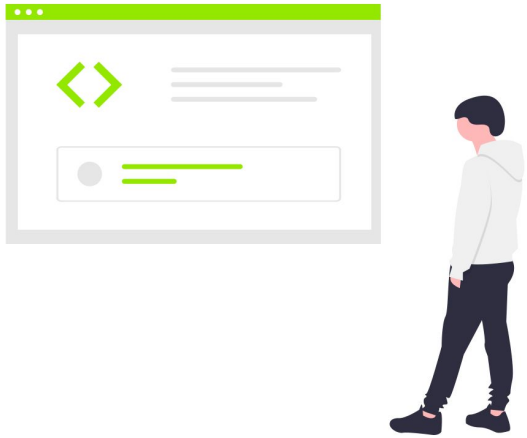
Ele armazena o endereço de memória da próxima instrução a ser buscada e executada pela CPU.



Multiprogramação

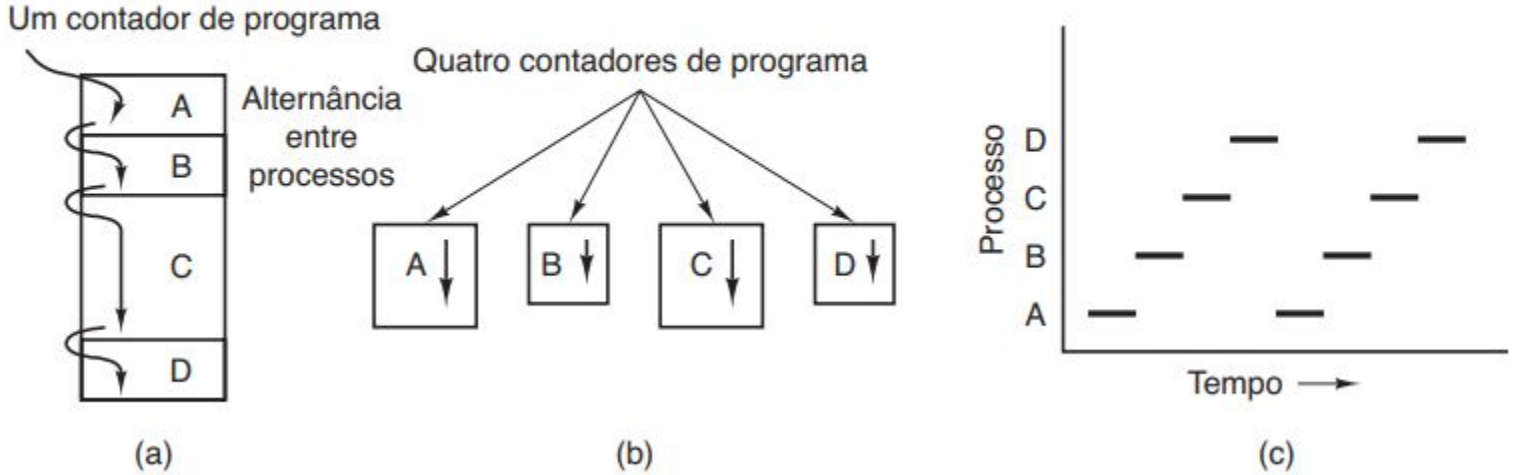
Conceitualmente, cada processo tem sua própria CPU virtual.

Na verdade, a CPU real troca a todo momento de processo em processo, mas, para compreender o sistema, é muito mais fácil pensar a respeito de uma coleção de processos sendo executados em (pseudo) paralelo.



Multiprogramação

(a) Multiprogramação de quatro programas. (b) Modelo conceitual de quatro processos sequenciais independentes. (c) Apenas um programa está ativo de cada vez.



Criação de processos

Quatro eventos principais fazem com que os processos sejam criados:

1. Inicialização do sistema.

2. Execução de uma chamada de sistema de criação de processo por um processo em execução.

3. Solicitação de um usuário para criar um novo processo.

4. Início de uma tarefa em lote.

Criação de processos

- Quando um sistema operacional é inicializado, em geral uma série de processos é criada.
- Alguns desses processos são de primeiro plano, isto é, processos que interagem com usuários (humanos) e realizam trabalho para eles.
- Outros operam no segundo plano e não estão associados com usuários em particular, mas em vez disso têm alguma função específica.

Término de processos

Cedo ou tarde, o novo processo terminará, normalmente devido a uma das condições a seguir:

1. Saída normal (voluntária).
2. Erro fatal (involuntário).
3. Saída por erro (voluntária).
4. Morto por outro processo (involuntário).

Estados de processos

1. Em execução (realmente usando a CPU naquele instante).
2. Pronto (executável, temporariamente parado para deixar outro processo ser executado).
3. Bloqueado (incapaz de ser executado até que algum evento externo aconteça).

Estados de processos

Transições entre esses estados ocorrem como mostrado.

Escalonador

É responsável por gerenciar a alocação de recursos do processador entre os vários processos em execução. Ele decide qual processo deve ser executado em um determ momento, visando otimizar o desempenho d sistema e garantir uma distribuição justa de recursos.

Escalonador

O escalonamento tem como principais objetivos:

- maximizar a u[lização do processador;
- maximizar o número de processos completados por unidade de tempo;
- garantir que todos os processos recebam o processador;
- minimizar o tempo de resposta para o usuário.

Escalonamento Não-Preemptivo

- O processo em execução não pode ser interrompido antes de sua conclusão ou liberação voluntária da CPU.
- A decisão de trocar de processo ocorre apenas quando o processo em execução termina ou entra em um estado de espera.

Escalonamento Preemptivo

- Permite que um processo em execução seja interrompido e retirado da CPU antes de sua conclusão.
- A decisão de trocar de processo pode ocorrer em momentos específicos, como no final de um quantum de tempo (Round Robin) ou quando um processo de alta prioridade chega.
- Oferece maior responsividade e flexibilidade, especialmente em ambientes multitarefa.

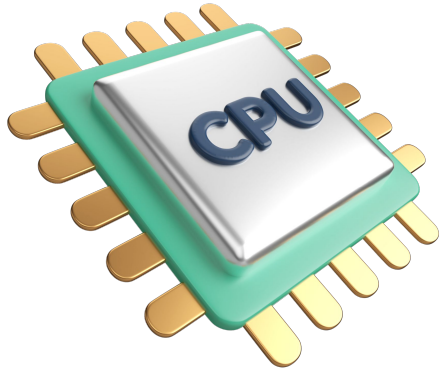
Tipos de processos

Os processos em execução, do usuário, podem assumir dois tipos diferentes, de acordo com suas características de uso de CPU e periféricos:

Tipos de processos

Processo CPU-bound

- É aquele processo que utiliza muito a CPU.
- Ele ganha uma fatia de tempo e a utiliza por inteiro, sem desperdiçar nenhum tempo.
- É o caso de programas científicos, de cálculo numérico, estatística, matemática, e também na área de simulação.
- Normalmente fazem pouca ou nenhuma entrada de dados, e muito processamento.



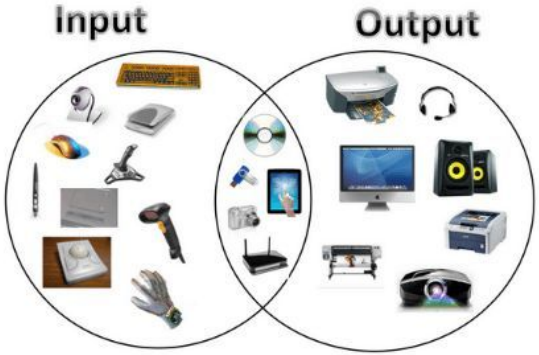
Tipos de processos

Processo I/O-bound

• é o tipo de processo que utiliza muito mais E/S do que CPU.

• Aplicações em Banco de Dados, onde se faz consultas e atualizações constantes em arquivos em disco são um bom exemplo deste tipo de processo.

• De acordo com essas características, podemos dizer que este tipo de processo permanece mais tempo em espera (tratando interrupções) do que propriamente em execução, ocupando a CPU por períodos mínimos de tempo.



Comunicação entre processos

Memória Compartilhada: Os processos podem compartilhar uma região de memória, permitindo a comunicação direta através da leitura e escrita nessa região compartilhada.

Comunicação entre processos

Filas de Mensagens: Os processos podem enviar mensagens uns aos outros por meio de filas de mensagens, onde as mensagens são armazenadas até que sejam lidas pelo destinatário.

Comunicação entre processos

Comunicação por Soquetes (Sockets): É um método de comunicação interprocesso que pode ser usado em redes locais ou pela internet. Processos podem se comunicar através de conexões de soquetes, usando protocolos como TCP/IP.

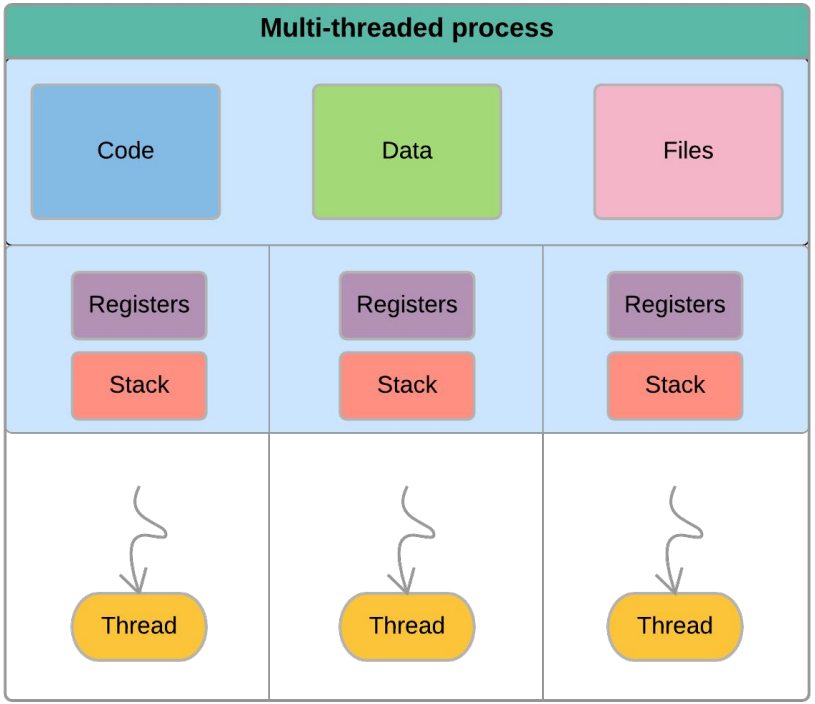
Thread ou processo leve

- Threads, ou processos leves, são unidades de execução em um programa.
- As threads são fluxos de um programa em execução. Um programa em execução é chamado de processo. Um processo, contém no mínimo uma thread.

Thread ou processo leve

- Cada thread compartilha o mesmo espaço de memória com outras threads do mesmo programa.
- Utilizadas para aumentar a eficiência e a capacidade de resposta dos programas, especialmente em sistemas com múltiplos núcleos de processamento.

Processos multithreads



Nota: A pilha de funções (stack) é uma área da memória que aloca dados/variáveis ou ponteiros quando uma função é chamada e desalocada quando a função termina.

Conceitos importantes antes de continuar

CONDIÇÃO DE CORRIDA

Quando mais de uma thread tenta acessar uma variável ao mesmo tempo temos uma condição de corrida. Ou seja, elas concorrem por algum recurso.

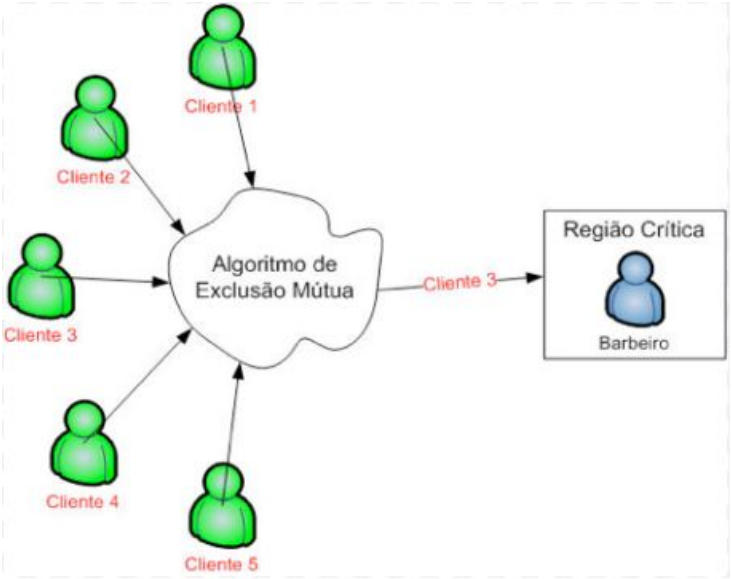
SEÇÃO CRÍTICA

As regiões críticas de um programa são as partes do código em que condições de corrida podem ocorrer.

Aquele trecho de código ou sequência de instruções que, ao serem escalonadas pelo sistema operacional, threads distintas podem estar executando de maneira concorrente.

EXCLUSÃO MÚTUA

A exclusão mútua é uma propriedade que garante que exclusivamente apenas uma unidade de execução (thread) esteja executando sua seção crítica do programa por vez.



ESPERA OCUPADA

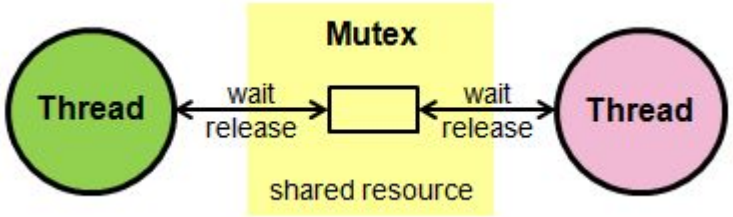
Enquanto uma thread está executando a seção crítica, outras threads que querem acessar o mesmo recurso devem aguardar. Quando essa espera é feita através de um loop, por exemplo, tentando acessar incessantemente o recurso, dizemos que é uma espera ocupada.

Formas de implementar exclusão mútua

Formas de implementar exclusão mútua

MUTEX

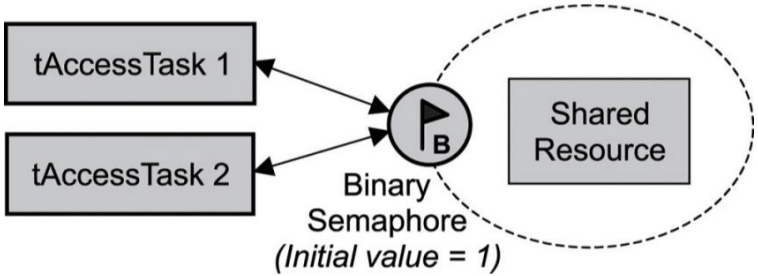
O mutex é uma implementação de exclusão mútua. Entretanto, ele gera espera ocupada. Isso faz com que, nem sempre, ele seja uma opção eficiente.



Formas de implementar exclusão mútua

Semáforos:

Semáforos são uma abstração usada para controlar o acesso concorrente a recursos compartilhados em um ambiente de programação concorrente ou paralela.



Formas de implementar exclusão mútua

Semáforos:

- Cada thread quando tentar executar sua seção crítica irá verificar antes se o semáforo permite o acesso ao recurso.
- Quando não está disponível, a thread dorme (wait state). Quando o recurso é liberado o sistema operacional sinaliza, através de eventos, para que o processo acorde e execute.

Formas de implementar exclusão mútua

Exemplo

Semáforos Binários:

- Pode ter apenas dois valores: 0 e 1.
- Usado para representar a disponibilidade ou ocupação de um recurso.
- Garante exclusão mútua, permitindo que apenas um processo por vez acesse o recurso.

Bibliografia

TANENBAUM, A. S. , **Sistemas Operacionais Modernos.** Segunda Edição, Prentice Hall, 2003.
Bibliografias Complementares. GALVIN, S., Operating System Concepts.