

Sistemas Operacionais I

Profa. Kalinka Regina Lucas Jaquie Castelo Branco
kalinka@icmc.usp.br

Universidade de São Paulo

Agosto de 2023

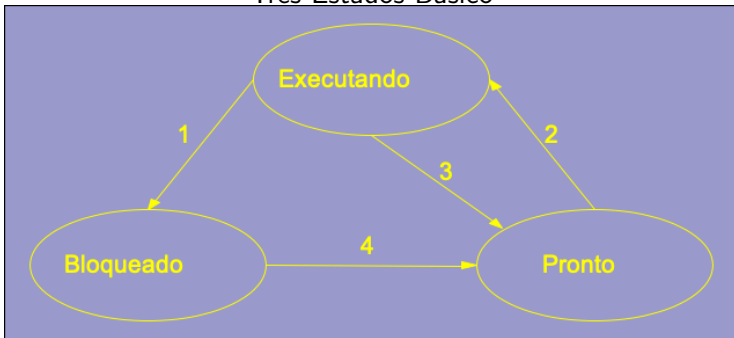
- Pai cria um processo filho, processo filho pode criar seus próprios processos
- Formação de hierarquia
 - Unix chama isto um "grupo de processos" ("process group").
- Windows não possui conceito de hierarquia de processos
 - Todos os processos são criados da mesma forma.

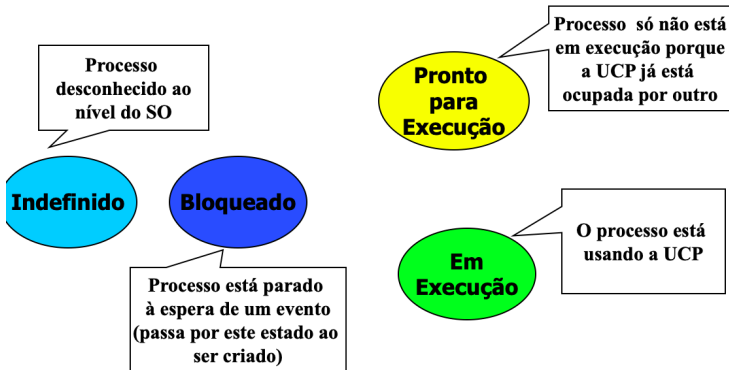
- Os processos do ponto de vista do S.O.
 - criação e remoção (destruição) de processos
 - controle do progresso dos processos
 - agir em condições excepcionais que acontecem durante a execução de um processo, incluindo interrupções e erros aritméticos
 - alocação de recursos de hardware entre os processos
 - fornece um meio de comunicação através de mensagens ou sinais entre os processos.

- Existem 2 tipos de sistemas, com relação aos processos.
- Sistemas Estáticos:
 - Geralmente são sistemas projetados para executar uma única aplicação. Permite que todos os processos necessários já estejam presentes quando o sistema é iniciado.
- Sistemas Dinâmicos
 - Possuem um número variável de processos. Necessita de uma forma de criar e destruir processos durante a execução.

- Os Estados de um processo:
 - Desde o momento em que um processo codificado pelo programador for colocado em memória ou disco, até o momento de sua destruição, ele poderá passar por quatro estados;
 - **Indefinido:** quando o processo é desconhecido ao S.O.. Um processo estará neste estado antes de ser criado e depois de ser destruído. (Neste ponto, ele será apenas um bloco de código no disco ou na memória).
 - **Bloqueado:** quando o processo estiver parado à espera da ocorrência de um evento, equivalente a não estar em andamento progressivo. Ao ser criado, o processo passará por este estado.
 - **Pronto para a execução:** quando o processo só não estiver em execução pelo fato da CPU estar sendo utilizada por outro processo.
 - **Em execução:** quando o processo estiver tendo andamento progressivo normal, usando a CPU.

Três Estados Básico





As ações provocam mudanças de estado. As ações responsáveis pelas transições tem o seguinte significado:

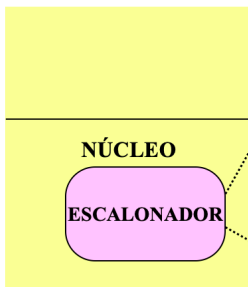
- **Criar:** colocar o processo a memória, tornando-o conhecido ao nível do sistema.
- **Acordar:** liberar o andamento do processo. Esta ação será desempenhada quando ocorrer algum evento ou o disparo inicial do processo, fazendo com que haja a transição do estado bloqueado para o pronto para execução.
- **Despachar:** dar sequência imediata ao andamento do processo. O processo deverá estar em uma fila, e a ocorrência dessa ação resultará na retirada do processo da fila e sua colocação no estado em execução.
- **Bloquear:** parar o andamento do processo para esperar a ocorrência de um evento. A ação bloquear será desempenhada quando uma determinada condição não for satisfeita, ou quando ocorrer o término do processo.
- **Preempção (Suspende):** parar o andamento do processo por motivos alheios ao mesmo, como acontece na comutação forçada de fatias de tempo. Esta ação retirará o processo da CPU e o colocará em uma fila, no estado pronto para a execução, liberando a utilização da CPU.
- **Destruir:** liberar a área de memória ocupada pelo processo, tornando-o desconhecido ao nível do sistema.

Processo - Ações do Escalonador

Sistemas Operacionais
I

Profa.
Kalinka
Branco

SISTEMA OPERACIONAL



- Processos CPU-bound (orientados à CPU): processos que utilizam muito o processador;
 - Tempo de execução é definido pelos ciclos de processador.
- Processos I/O-bound (orientados à E/S): processos que realizam muito E/S;
 - Tempo de execução é definido pela duração das operações de E/S.
- **IDEAL:** existir um balanceamento entre processos CPU-bound e I/O-bound.

- Outras características dos processos
 - Os processos não podem ser programados com suposições embutidas sobre temporização
 - Em geral, um sistema multiplexa um só processador central entre vários processos. Os instantes em que o processador é realocado de um processo para outro são, em geral, imprevisíveis.
 - Um algoritmo de **escalonamento** determina quando parar o trabalho com um processo e atender um diferente.

- Escalonador de processos escolhe o processo que será executado pelo CPU.
- Escalonamento é realizado com auxílio do hardware.
- Escalonador deve se preocupar com a eficiência da CPU, pois o chaveamento de processos é complexo e custo - afetando desempenho do sistema e satisfação do usuário.
- Escalonador de processo é um processo que deve ser executado quando da **mudança de contexto** (troca de processo).

Mudança de Contexto

- *Overhead* de tempo
- Tarefa cara:
 - Salvar as informações do processo que está deixando a CPU em seu BCP - conteúdo dos registradores.
 - Carregar as informações do processo que será colocado na CPU - copiar do BCP o conteúdo dos registradores.

Antes da Mudança de Contexto

BCP-P2

PC = 0BF4h

PID = 2

Estado = pronto

Próximo processo

BCP-P4

PC = 074Fh

PID = 4

Estado = executando

PC = 074Fh

CPU

Depois da Mudança de Contexto

BCP-P2

PC = 0BF4h

PID = 2

Estado = executando

BCP-P4

PC = 074Fh

PID = 4

Estado = pronto

PC = 0BF4h

CPU

Situações nas quais escalonamento é necessário

- Um novo processo é criado;
- Um processo terminou sua execução e um processo pronto deve ser executado;
- Quando um processo é bloqueado (semáforo, dependência de E/S), outro deve ser executado;
- Quando uma interrupção de E/S ocorre o escalonador deve decidir por: executar o processo que estava esperando esse evento; continuar executando o processo que já estava sendo executado ou executar um terceiro processo que esteja pronto para ser executado;

- Tempo de execução de um processo é imprevisível:
 - CPU gera interrupções em intervalos entre 50 a 60 hz (ocorrências por segundo).
- Algoritmos de escalonamento podem ser divididos em duas categorias dependendo de como essas interrupções são tratadas:
 - **Preemptivo:** estratégia de suspender o processo sendo executado;
 - **Não-preemptivo:** estratégia de permitir que o processo sendo executado continue sendo executado até ser bloqueado por alguma razão (semáforos, operações de E/S-interrupção).

Categorias de Ambientes

- **Sistemas em Batch:** usuários não esperam por respostas rápidas; algoritmos preemptivos ou não-preemptivos;
- **Sistemas Interativos:** interação constante do usuário; algoritmos preemptivos; Processo interativo - espera comando e executa comando;
- **Sistemas em Tempo Real:** processos são executados mais rapidamente; tempo é crucial - sistemas críticos.

Características de algoritmos de escalonamento (qualquer sistema):

- **Justiça (Fairness):** cada processo deve receber uma parcela justa de tempo da CPU;
- **Balanceamento:** diminuir a ociosidade do sistema;
- **Políticas do sistema:** prioridade de processos.

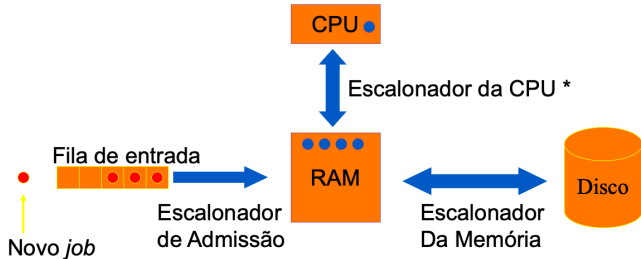
Características de algoritmos de escalonamento:

- Sistemas Batch
 - **Taxa de execução (throughput):** máximo número de jobs executados por hora;
 - **Turnaround time (tempo de retorno):** tempo no qual o processo espera para ser finalizado;
 - **Tempo de espera:** tempo gasto na fila de prontos;
 - **Eficiência:** CPU deve estar 100% do tempo ocupada.
- Sistemas Interativos
 - **Tempo de resposta:** tempo esperando para iniciar execução;
 - **Satisfação do usuários.**

Características de algoritmos de escalonamento:

- Sistemas de Tempo Real
 - **Prevenir perda de dados;**
 - **Previsibilidade:** prevenir perda da qualidade dos serviços oferecidos

Escalonamento *Three Level (Batch)*



- **Escalonador de admissão:** processos menores primeiro; processos com menor tempo de acesso à CPU e maior tempo de interação com dispositivos de E/S;
- **Escalonador da Memória:** decisões sobre quais processos vão para a MP:
 - A quanto tempo o processo está esperando?
 - Quanto tempo da CPU o processo já utilizou?
 - Qual o tamanho do processo?
 - Qual a importância do processo?
- **Escalonador da CPU:** seleciona qual o próximo processo a ser executado.

Algoritmos para Sistemas *Batch*

- *First-Come First-Served* (ou FIFO);
- *Shortest Job First* (SJF);
- *Shortest Remaining Time Next* (SRTN).

First-Come First-Served

- Não-preemptivo;
- Processos são executados na CPU seguindo a ordem de requisição;
- Fácil de entender e programar;
- Desvantagem - Ineficiente quando se tem processos que demoram na sua execução.

First-Come First-Served

Shortest Job First

- Não-preemptivo;
- Possível prever o tempo de execução do processo;
- Menor processo é executado primeiro;
- Menor *turnaround*;
- Desvantagem - Baixo aproveitamento quando se tem poucos processos prontos para serem executados.

Shortest Job First

A →	a
<u>B</u> →	<u>b+a</u>
C →	<u>c+b+a</u>
<u>D</u> →	<u>d+c+b+a</u>

Tempo médio-*turnaround* $(4a+3b+2c+d)/4$

Contribuição → se $a < \underline{b} < \underline{c} < \underline{d}$ tem-se o mínimo tempo médio.

Shortest Job First

8	4	4	4
A	B	C	D

Em ordem:

Turnaround A = 8

Turnaround B = 12

Turnaround C = 16

Turnaround D = 20

Média $\rightarrow 56/4 = 14$

4	4	4	8
B	C	D	A

Menor *job* primeiro:

Turnaround B = 4

Turnaround C = 8

Turnaround D = 12

Turnaround A = 20

Média $\rightarrow 44/4 = 11$

$(4a+3b+2c+d)/4$	← Número de Processos
------------------	-----------------------

Shortest Remaining Time Next

- Preemptivo;
- Processos com menor tempo de execução são executados primeiro;
- Se um processo novo chega e seu tempo de execução é menor do que do processo corrente na CPU, a CPU suspende o processo corrente e executa o processo que acabou de chegar;
- **Desvantagem:** processos que consomem mais tempo podem demorar muito para serem finalizados se muitos processos pequenos chegarem!

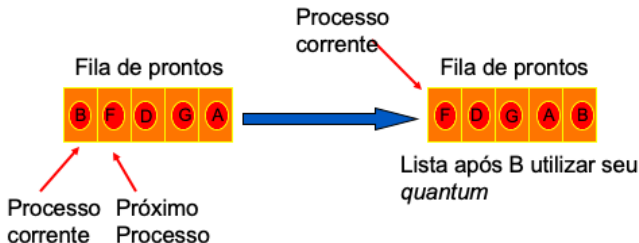
- *Round-Robin*;
- Prioridade;
- Múltiplas Filas;
- *Shortest Process Next*;
- Garantido;
- *Lottery*;
- *Fair-Share*.

Utilizam escalonamento em dois níveis: Escalonador de CPU e Memória!!

Round-Robin

- Antigo, mais simples e mais utilizado;
- Preemptivo;
- Cada processo recebe um tempo de execução chamado quantum; ao final desse tempo, o processo é suspenso e outro processo é colocado em execução;
- Escalonador mantém uma lista de processos prontos.

Round-Robin



Round-Robin

- Tempo de chaveamento de processos;
- **quantum:** se for muito pequeno, ocorrem muitas trocas diminuindo, assim, a eficiência da CPU; se for muito longo o tempo de resposta é comprometido;

Round-Robin

Exemplos:

$\Delta t = 4 \text{ mseg}$

$x = 1 \text{ mseg}$

quantum

→ 25% de tempo de CPU é
perdido → menor eficiência

$\Delta t = 100 \text{ mseg}$

$x = 1 \text{ mseg}$

→ 1% de tempo de CPU é
perdido → Tempo de espera dos processos é
maior

chaveamento

Round-Robin

Exemplos:

$$\Delta t = 4 \text{ mseg}$$

$x = 1 \text{ mseg}$ → 25% de tempo de CPU é
perdido → menor eficiência

$$\Delta t = 100 \text{ mseg}$$

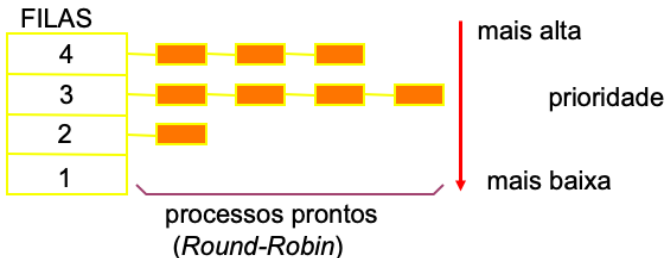
$x = 1 \text{ mseg}$ → 1% de tempo de CPU é
perdido → Tempo de espera dos processos é
maior

quantum razoável: 20-50 mseg

Algoritmo com Prioridades

- Cada processo possui uma prioridade - os processos prontos com maior prioridade são executados primeiro.
- Prioridades são atribuídas dinâmica ou estaticamente.
- Classes do processos com mesma prioridade.
- Preemptivo.

Algoritmo com Prioridades



Algoritmo com Prioridades

- Como evitar que os processos com maior prioridade sejam executado indefinidamente?
 - Diminuir a prioridade do processo corrente e trocá-lo pelo próximo processo com maior prioridade (chaveamento)
 - Cada processo possui um *quantum*

Múltiplas Filas

- CTSS (*Compatible Time Sharing System*)
- Classes de prioridades
- Cada classe de prioridades possui *quanta* diferentes
- Assim, a cada vez que um processo é executado e suspenso ele recebe mais tempo para execução
- Preemptivo

Múltiplas Filas

- Ex. Um processo precisa de 100 *quanta* para ser executado
 - Inicialmente, ele recebe um *quantum* para execução
 - Das próximas vezes ele recebe, respectivamente, 2, 4, 8, 16, 32 e 64 *quanta* (7 chaveamentos) para execução
 - Quanto mais próximo de ser finalizado menos frequente é o processo na CPU - eficiência

Algoritmos Shortest Process Next

- Mesma ideia do *Shortest Job First*
- Processos Interativos: não se conhece o tempo necessário para execução
- Como empregar esse algoritmo: ESTIMATIVA de TEMPO!

Outros Algoritmos

- Algoritmo Garantido:
 - Garantias são dadas aos processos dos usuários (n usuários - $1/n$ do tempo de CPU para cada)
- Algoritmo *Loterry*
 - Cada processo recebe *tickets* que lhe dão direito de execução
- Algoritmos *Fair-Share*
 - O dono do processo é levado em conta
 - Se um usuário A possui mais processos que um usuário B, o usuário A terá prioridade no uso da CPU

Processo - Escalonamento (*Sistemas em Tempo Real*)

Sistemas
Operacionais
I

Profa.
Kalinka
Branco

- Tempo é um fator crítico
- Sistema Crítico
 - Aviões
 - Hospitais
 - Usinas Nucleares
 - Bancos
 - Multimídia
- Ponto importante: obter respostas em atraso é tão ruim quanto não obter respostas.

Processo - Escalonamento (*Sistemas em Tempo Real*)

Sistemas
Operacionais
I

Profa.
Kalinka
Branco

- Tipos de STR
 - *Hard Real Time*: atrasos não são tolerados (aviões, usinas nucleares, hospitais)
 - *Soft Real Time*: Atrasos são tolerados (Bancos, Multimídia)
- Programas são divididos em vários processos
- Eventos causam execução de processos
 - **Periódicos**: Ocorrem em intervalos de tempos regulares
 - **Aperiódicos**: Ocorrem em intervalos de tempo irregulares

Processo - Escalonamento (*Sistemas em Tempo Real*)

Sistemas
Operacionais
I

Profa.
Kalinka
Branco

- Algoritmos podem ser estáticos ou dinâmicos
 - **Estáticos:** decisões de escalonamento antes do sistema começar (informações disponíveis previamente)
 - **Dinâmicos:** decisões de escalonamento em tempo de execução

Continuemos com o **THREADS**