


# Capítulo 5: Escalonamento da CPU


**SUMÁRIO:**

- Conceitos básicos
- Critérios de escalonamento
- Algoritmos de escalonamento
- Escalonamento multi-processador
- Escalonamento em tempo real



Operating System Concepts 5.1 Silberschatz, Galvin and Gagne ©2002

1



## Conceitos básicos

Objectivo do escalonamento do processador (principal recurso):

**É maximizar a utilização da CPU via multiprogramação.**


Ideia base da multiprogramação:

**Quando um processo tem de esperar (p.ex operação I/O), o sistema operativo retira-lhe a CPU, dando-a a outro processo.**

Conceitos básicos:

- Ciclo "burst CPU"- "I/O" – execução dum processo consiste dum ciclo de execução na CPU e espera I/O
- Escalonador da CPU
- Escalonamento preemptivo
- **Despachador**

É o módulo que despacha o controlo da CPU para o processo selecionado pelo escalonador de curto-prazo.



Operating System Concepts 5.2 Silberschatz, Galvin and Gagne ©2002

2

## Sequência alternada de intermitências de CPU e I/O

- O sucesso do escalonamento da CPU depende da seguinte **observação** sobre os processos:
 

**A execução dum processo consiste em vários ciclos CPU-I/O tal que uma intermitência (*burst*) de execução da CPU alterna com uma intermitência (*burst*) de espera pela finalização duma operação I/O.**

Histograma de CPU bursts

Os processos *I/O-bound* têm em geral um grande número de *CPU bursts* de curta duração.  
Os processos *CPU-bound* têm em geral um pequeno número de *CPU-bursts* de longa duração.

Operating System Concepts 5.3 Silberschatz, Galvin and Gagne ©2002

3

## Escalonador da CPU


- Sempre que a CPU fica livre, cabe ao sistema operativo **selecionar** um dos processos da **ready queue** a fim de o colocar em execução.
- A **seleção** é efetuada pelo escalonador de curto prazo ou escalonador da CPU.
- O escalonamento da CPU pode ter lugar quando um processo:
  - Comuta do estado **RUNNING** para os estados
    - i. **WAITING** , ii. **Ready** ou iii. **Terminated**
  - iv. Comuta do estado **WAITING** para o estado **READY**.

Ready queue é uma abstracção ..  
Pode ser fifo, fila de prioridades, árvore, lista-ligada simples etc.

**Situações geradoras de escalonamento da CPU**

Operating System Concepts ... agne ©2002

4



## Escalonamento Não-Preemptivo

O processo ocupa a CPU até ao seu **término** ou até que passe ao estado waiting .. as 1ª e 4ª situações.


Agora um **novo processo** da "ready queue" tem que ser selecionado para execução.

O escalonamento é dito ser não-preemptivo

MS Windows 3.1, MAC OS (antes de OSX), SO's especializadas.


Os sistemas operativos não-preemptivos não são adequados para sistemas de tempo real, pois não garantem a execução em primeiro lugar dos processos com prioridade mais alta.

Cooperative Multitasking ? "one poorly designed program can consume all of the CPU time for itself or cause the whole system to hang".



Operating System Concepts 5.5 Silberschatz, Galvin and Gagne ©2002

5



## Escalonamento Preemptivo

Preemptive multitasking involves the use of an interrupt mechanism which suspends the currently executing process and invokes a scheduler to determine which process should execute next.

Preempção nas 2ª e 3ª situações.

Necessite algum Hardware/Mecanismo específico (Hardware Timer Interrupt Lines)


Mas aqui há um custo a pagar. Considere que dois processos partilham dados. Um dos processos está a atualizar os dados quando ocorre a preempção do segundo processo que passa a executar e a ler os dados.

- Os dados assim podem ficar num estado inconsistente.
- Sincronização ..estudada mais tarde.

Algumas tarefas (do OS) não são "interruptíveis"  
p.ex o próprio "interrupt handler routine"


As decisões de escalonamento da CPU têm lugar nas 4 seguintes circunstâncias:

1. um processo comuta do estado running para o estado waiting.
  - Interrupção I/O ou Sleep
  - Chamado ao sistema **wait()** espera a terminação dum processo filho.
2. um processo comuta do estado running para o estado ready
  - Ocorrência dum interrupção
3. um processo comuta do estado waiting para o estado ready
  - Terminação de I/O
  - Recurso livre
4. Um processo termina
  - System call **\_exit()**



Operating System Concepts 5.6 Silberschatz, Galvin and Gagne ©2002


6




## Despachador

- ❑ É o módulo que despacha o controlo da CPU para o processo seleccionado pelo escalonador de curto-prazo.
- ❑ Executa as seguintes operações :
  - ❑ comutação de contexto
  - ❑ comutação para o modo de utilizador
  - ❑ salto para o endereço certo de memória do programa por forma a (re-)executá-lo
- ❑ O despacho deve ser tão rápido quanto possível.
- ❑ O tempo que decorre entre a paragem de execução dum processo e o início doutro é designado por **latência de despacho**.

Operating System Concepts 5.7 Silberschatz, Galvin and Gagne ©2002



7




## Critérios de Escalonamento

Há vários critérios para comparar algoritmos de escalonamento:

- ❑ **Utilização da CPU:** maximizar a utilização da CPU. Deve variar entre 40% e 90% em sistemas de tempo real. Um critério de maximização.
- ❑ **Débito (throughput):** maximizar o nº de processos concluídos por unidade de tempo. Critério de maximização.
- ❑ **Tempo de circulação (turnaround):** tempo que decorre entre o instante em que um processo é submetido e o instante em que é concluído. Critério de minimização.
- ❑ **Tempo de espera:** é a soma dos períodos dispendidos na ready queue. Critério de minimização.
- ❑ **Tempo de resposta:** minimizar o tempo que decorre entre a submissão dum pedido e o início da resposta. Este critério é adequado para sistemas interactivos. Critério de minimização.

**geralmente usamos uma média**

Operating System Concepts 5.8 Silberschatz, Galvin and Gagne ©2002



8



## Algoritmos de Escalonamento

- ❑ First-Come, First-Served (FCFS)
- ❑ Shortest-Job-First (SJT)
- ❑ Prioridade
- ❑ Round-Robin (R-R)
- ❑ Multi-fila
- ❑ Multi-fila com transbordo

Metricos :


TME Tempo Médio de Espera

TMT Tempo Médio de Turnaround



Operating System Concepts 5.9 Silberschatz, Galvin and Gagne ©2002

9

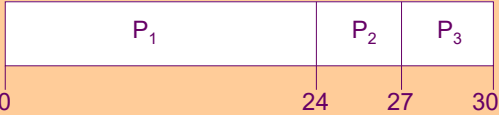


## First-Come, First-Served (FCFS)


- ❑ O algoritmo mais simples: processos são seleccionados ou *servidos* pela ordem de chegada à ready queue.
- ❑ Assim que a CPU é libertada, o processo à cabeça da *ready queue* é seleccionado e despachado para a CPU.

Processo	Burst Time
$P_1$	24
$P_2$	3
$P_3$	3

- ❑ Suponha que os processos chegam pela ordem:  $P_1, P_2, P_3$   
A Carta Gantt de escalonamento é:



- ❑ Tempos de espera:  $P_1 = 0; P_2 = 24; P_3 = 27$
- ❑ TME:  $(0 + 24 + 27)/3 = 17$



Operating System Concepts 5.10 Silberschatz, Galvin and Gagne ©2002

10

## FCFS (Cont.)


- Suponha que os processos chegam pela ordem:  $P_2, P_3, P_1$ .
- A Carta de Gantt de escalonamento é:



- Tempos de espera  $P_1 = 6; P_2 = 0; P_3 = 3$
- TME:  $(6 + 0 + 3)/3 = 3$ 
  - Muito melhor que o caso anterior.
  - *Convoy effect*: processo curto antes de processo longo


## Conclusões FCFS

- O tempo médio de espera é, por vezes, bastante elevado, mas isto depende muito da duração e frequência dos bursts.
- O algoritmo FCFS não é preemptivo. Portanto não é adequado para sistemas interativos (time sharing) ou de tempo real.
- Batch Systems – Pode ser Adequado




## Shortest-Job-First (SJF)

- Associa-se a cada processo (ao PCB) o tempo do seu próximo CPU burst.
- Usa-se estes tempos para escalonar/seleccionar o processo com o CPU burst mais pequeno.
- Quando dois processo têm o mesmo CPU burst, o desempate faz-se por FCFS.
- Dois esquemas:
  - **não-preemptivo** – uma vez a CPU é atribuída a um processo, este não pode ser preemptado até completar o seu CPU burst.
  - **preemptivo** – se um novo processo chega à ready queue com um CPU burst menor que o tempo restante do processo em execução, então há preempção. Este esquema é conhecido por Shortest-Remaining-Time-First (SRTF).
- SJF é **ótimo** – uma vez que minimiza o tempo médio de espera dum dado conjunto de processos.
- O problema está em **determinar** qual é o valor do próximo CPU burst dum processo.



Operating System Concepts 5.13 Silberschatz, Galvin and Gagne ©2002

13



## Exemplos


### SJF não-preemptivo

Processo	Arrival Time	Burst Time
$P_1$	0.0	7
$P_2$	2.0	4
$P_3$	4.0	1
$P_4$	5.0	4

- SJF (não-preemptivo)

0      3      7    8      12    16

- Tempo médio de espera  
 $(0 + 6 + 3 + 7)/4 = 4$



Operating System Concepts 5.14 Silberschatz, Galvin and Gagne ©2002

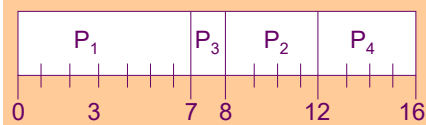
14

## Exemplos

### SJF não-preemptivo

Processo	Arrival Time	Burst Time
$P_1$	0.0	7
$P_2$	2.0	4
$P_3$	4.0	1
$P_4$	5.0	4

#### SJF (não-preemptivo)

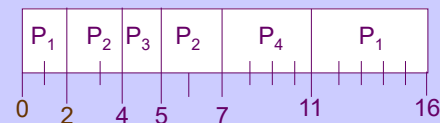


Tempo médio de espera  
 $(0 + 6 + 3 + 7)/4 = 4$

### SJF preemptivo

Processo	Arrival Time	Burst Time
$P_1$	0.0	7
$P_2$	2.0	4
$P_3$	4.0	1
$P_4$	5.0	4

#### SJF (preemptivo)



Tempo médio de espera  
 $(9 + 1 + 0 + 2)/4 = 3$

## Determinição da duração do Proximo CPU Burst

- Apenas uma estimativa é possível
- Utilize-se uma média móvel baseado na duração de CPU bursts anteriores

- $t_n$  = duração verdadeira do  $n^{\text{th}}$  CPU burst
- $\tau_{n+1}$  = valor estimado para o proximo CPU burst
- $\alpha, 0 \leq \alpha \leq 1$
- Define: 
$$\tau_{n+1} = \alpha t_n + (1 - \alpha)\tau_n.$$

Valor típico de  $\alpha=0.5$ .

Os bancos e Euribor



## Previsão da duração do proximo CPU Burst

CPU burst ( $t_j$ )	6	4	6	4	13	13	13	...	
"guess" ( $\tau_j$ )	10	8	6	6	5	9	11	12	...

Implementação ? +PCB

Operating System Concepts 5.17 Silberschatz, Galvin and Gagne ©2002

17

## Escalonamento por prioridades

- O algoritmo SJF é um caso particular do algoritmo de escalonamento por prioridades, em que a prioridade é o próximo tempo previsível de CPU burst.
- Uma prioridade é atribuída a cada processo, e o escalonador atribui a CPU ao processo com maior prioridade (menor inteiro).
  - preemptivo
  - não-preemptivo
- Se dois processos têm a mesma prioridade, o desempate é feito recorrendo ao FCFS
- **Problema** ≡ inanição (starvation) – processos de baixa prioridade arriscam-se a nunca executar.
- **Solução** ≡ envelhecimento (aging) – à medida que o tempo passa, a prioridade dum processo aumenta.

Processo	Prioridade	Burst Time
$P_1$	3	10
$P_2$	1	1
$P_3$	3	2
$P_4$	4	1
$P_5$	2	5

□ não-preemptivo

□ Tempo médio de espera = 8.2

Operating System Concepts 5.18 Silberschatz, Galvin and Gagne ©2002

18

## Critérios de prioridade

A prioridade atribuída a um processo pode ser definida em função dos seguintes factores:

Figure 2.1: Preemptive Priority-Based Scheduling behaviour example.

**Factores internos:**

- limites de tempo
- requisitos de memória
- nº de ficheiros abertos
- duração média dos bursts de I/O
- duração média dos bursts de CPU

**Factores externos:**

- importância do processo
- preço pago pela utilização
- proprietário do processo

Operating System Concepts 5.19 Silberschatz, Galvin and Gagne ©2002

19

## Round Robin (RR)

- Este algoritmo foi concebido para sistemas de *time-sharing*.
- É semelhante ao FCFS, mas é preemptivo.
- Cada processo obtém uma pequena unidade de tempo na CPU (*time quantum* ou *time slice*), vulgarmente 10-100 milisegundos. Após decorrer este tempo, o processo é preemptado e adicionado à cauda da fila READY. A fila READY é tratada como uma fila circular.
- Se há  $n$  processos na fila READY e o time quantum é  $q$ , então cada processo obtém  $1/n$  do tempo da CPU em fatias de  $q$  unidades de tempo duma vez. Nenhum processo espera mais do que  $(n-1)q$  unidades de tempo.
- Desempenho
  - $q$  grande  $\Rightarrow$  FIFO
  - $q$  pequeno  $\Rightarrow$   $q$  tem de ser grande relativamente à comutação de contexto; caso contrário, a sobrecarga é muito elevada.

**time quantum = 20 ms**

Process	Burst Time
$P_1$	53
$P_2$	17
$P_3$	68
$P_4$	24

□ A Carta Gantt é:

$P_1$	$P_2$	$P_3$	$P_4$	$P_1$	$P_3$	$P_4$	$P_1$	$P_3$	$P_3$	
0	20	37	57	77	97	117	121	134	154	162

□ Tipicamente, turnaround médio é mais elevado do que SJF, mas tem melhor resposta.

Operating System Concepts 5.20 Silberschatz, Galvin and Gagne ©2002

20

## Comparação de tempos de processamento

**time quantum versus tempo de comutação de contexto**

process time = 10

quantum	context switches
12	0
6	1
1	9

**tempo de turnaround varia com o time quantum**

process	time
$P_1$	6
$P_2$	3
$P_3$	1
$P_4$	7

Operating System Concepts 5.21 Silberschatz, Galvin and Gagne ©2002

21

## Algorithm Evaluation

**(1) Modelação e Simulação**

Define-se um workload/trace e depois simule-se o desempenho de cada algoritmo para este workload/trace.

Workloads podem ser :

- Determinística – Define um workload baseado por exemplo em dados reais ou inventados
- Aleatório – utilizando processos aleatórios e probabilísticos  
tempo de chegada .. "poisson"  
burst time .. "exponencial"

**(2) Analise Matemática : Queueing models M/M/1 etc.**

Operating System Concepts 5.22 Silberschatz, Galvin and Gagne ©2002

22

## Fila multi-nível

- Este tipo de escalonamento é usado quando é fácil classificar os processos em classes distintas (processos interactivos, processos batch, etc.).
- A fila READY é particionada em várias filas, uma por cada classe de processos foreground (interactive) background (batch)
- Cada fila tem o seu próprio algoritmo de escalonamento: foreground – RR background – FCFS
- O escalonamento entre as filas tem de ser feito.
  - Escalonamento de prioridades fixas; (i.e. serve todas as filas, desde as foreground até às background). Problema: inanição.
  - Time slice – cada fila obtém uma certa quantidade de tempo da CPU que pode ser escalonado pelos seus processos; por exemplo, 80% para foreground em RR 20% para background em FCFS

**escalonamento de fila multi-nível**

Operating System Concepts 5.23 Silberschatz, Galvin and Gagne ©2002

23

## Fila multi-nível com transbordo

- Um processo pode mover-se entre várias filas; a técnica de envelhecimento pode ser implementada desta forma.
- Outras características:
  - prioridades por fila
  - preempção
  - generalidade
  - configurabilidade

- Três filas:
  - $Q_0$  – time quantum de 8 milisegundos
  - $Q_1$  – time quantum de 16 milisegundos
  - $Q_2$  – FCFS
- Escalonamento
  - Um novo processo entra na fila  $Q_0$ , a qual segue uma política FCFS. Quando ganha a CPU, o processo recebe 8 ms. Se não termina em 8 ms, o processo é trasladado para a fila  $Q_1$ .
  - Em  $Q_1$ , o processo é servido novamente por uma política de escalonamento FCFS e recebe 16 ms adicionais. Se mesmo assim não termina, o processo é preempcionado e trasladado para a fila  $Q_2$ .

Operating System Concepts 5.24 Silberschatz, Galvin and Gagne ©2002

24

## Escalonamento Multi-Processor

- ❓ Não existe uma solução óptima de escalonamento mesmo para sistemas uniprocessador.
- ❓ O problema do escalonamento torna-se ainda mais complexo para sistemas multiprocessador.
- ❓ *Processadores homogéneos* dentro do sistema multi-processorador.
- ❓ *Partilha de carga*
- ❓ *Multi-processamento assimétrico* – só um processador acede às estruturas de dados do sistema, aliviando a necessidade de partilha de dados.
- ❓ É usada uma **única fila ready**, e não uma fila por processador, para evitar que haja algum processador inactivo enquanto outros têm processos na suas filas ready à espera.

- **Há duas políticas de escalonamento multiprocessador:**
  - **Processadores auto-escalonáveis.** Neste caso, cada processador é responsável pela selecção dum processo existente na fila ready partilhada.
  - **Processador mestre - processador escravo.** Há um processador (mestre) que desempenha o papel de escalonador dos restantes (escravos)

Cf a Caixa-Geral e McDonalds !!

Operating System Concepts
5.25
Silberschatz, Galvin and Gagne ©2002

25

## Escalonamento em tempo-real

Dois tipos de sistemas operativos de tempo real:

- ❓ **sistemas estritos de tempo real** (*hard real-time systems*). São necessários para garantir a conclusão duma tarefa crítica dentro duma quantidade de tempo pré-definida.
- ❓ **sistemas latos de tempo real** (*soft real-time systems*). São menos restritivos. Mas, os processos críticos têm sempre a máxima prioridade.

**Escalonamento:**

- **Escalonamento por prioridades.** Processos de tempo real têm prioridade máxima.
- **Manutenção da prioridade.** Ao contrário doutros processos, um processo de tempo real mantém a sua prioridade.
- **Latência de despacho.** Deve ser baixa o mais possível. Para isso, há sistemas operativos que admitem a preempção das "chamadas ao sistema" de longa duração.

The diagram illustrates the response interval for a real-time system. It starts with an event, followed by interrupt processing. Once the process is made available, there is a period of dispatch latency. This is followed by a period of conflicts, then a dispatch, and finally the real-time process execution. The total time from event to response to event is the response interval.

Operating System Concepts
5.26
Silberschatz, Galvin and Gagne ©2002

26