

Fundamentos de Sistemas de Operação

Unix Windows NT Netware Mac OS DOS/V/S Vax/VMS
Linux Solaris HP/UX AIX Mach Chorus

*(Revisitando o) Escalonamento do CPU:
...uma perspectiva histórica.*

Um mundo simples... (1)

□ Assumindo que

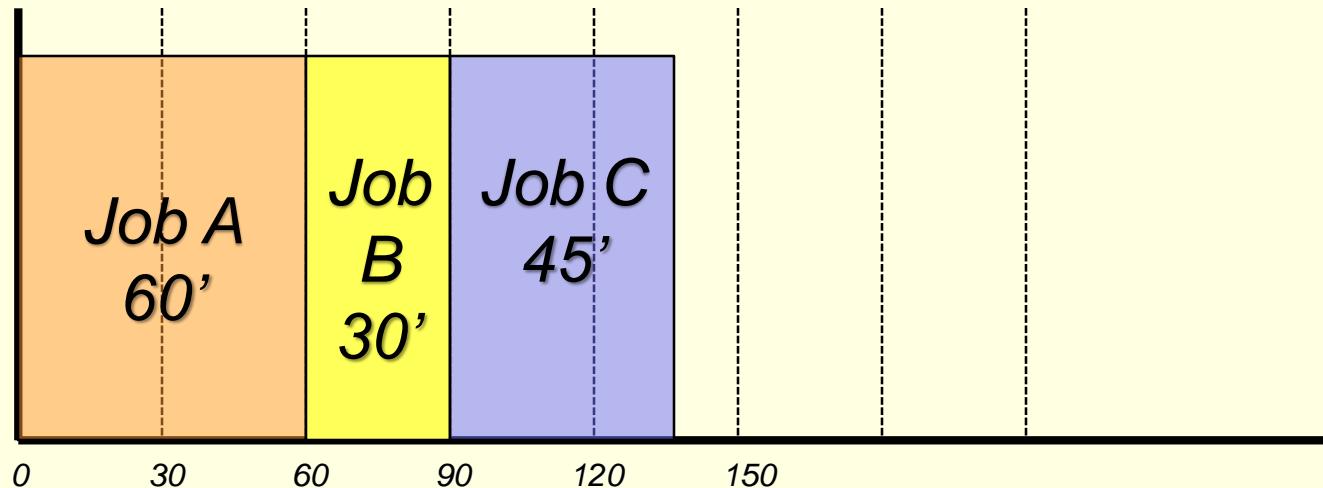
- *Conhecemos previamente (oráculo?) o tempo de execução de um “job” (a.k.a. processo ou cadeia de processos)*
- *Os “jobs” vão chegando para serem executados e, quando em execução, não executam operações de I/O (i.e., só gastam CPU)*

□ E que

- *O escalonamento é justo (não passam à frente jobs que chegaram “atrasados”) ☺*
- *Como podemos calcular o tempo de turnaround (tempo de espera pelo resultado de uma execução)?*

Um mundo simples... (2)

- Turnaround = $T_{fim} - T_{chegada} = \text{atraso} + (T_{fim} - T_{início})$



$$\text{Average Turnaround} = (\sum T_{fim}(i) - T_{chegada}(i)) / N^{\circ} \text{ jobs}$$

(se chegaram todos ao mesmo tempo) = $(60+90+135)/3 = 101.7$

Turnaround

□ Cenário

- Às 9h da manhã, tudo em fila para entrar na loja
- No LCD está um aviso: **Average Turnaround = 101.7**
 - Em média, as pessoas que aqui estão, daqui a 101,7 minutos têm os seus assuntos despachados e podem ir-se embora ☺

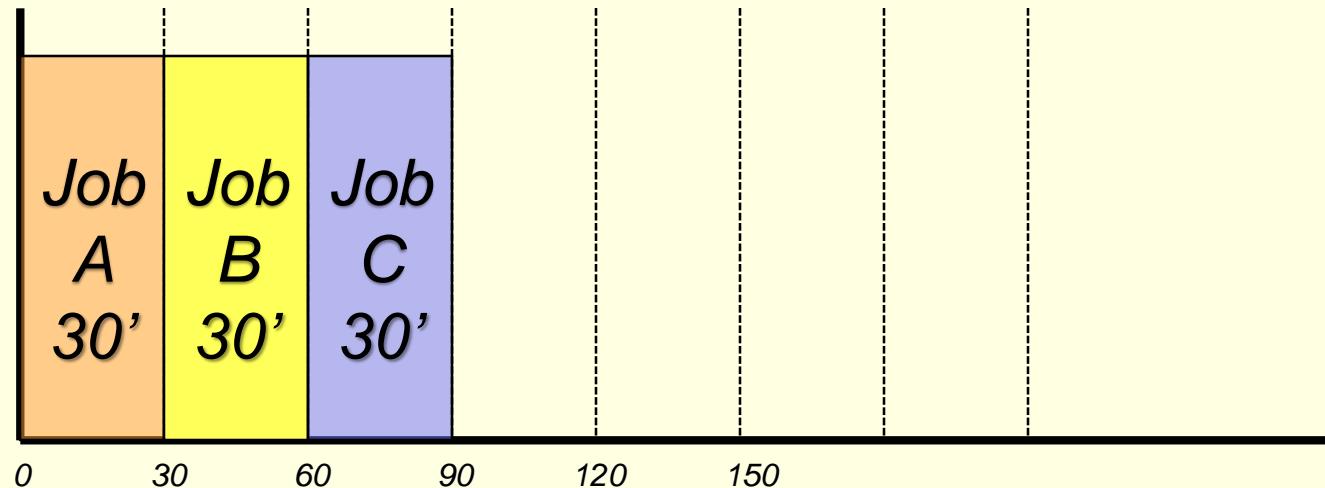
Políticas de Escalonamento... (1)

□ *Políticas de Escalonamento (do CPU)*

- *Com base em parâmetros / características das tarefas a executar, tomam as decisões (e.g., quando é executado)*
- *FIFO ou FCFS (First Come, First Served)*
 - O escalonamento é justo (*não passam à frente jobs que chegaram “atrasados”*) ☺
 - *Não favorece o Turnaround*
- *SJF (Shortest Job First)*
 - *Jobs de curta duração são favorecidos, os utilizadores agradecem ☺*

Políticas de Escalonamento... (2)

□ FIFO

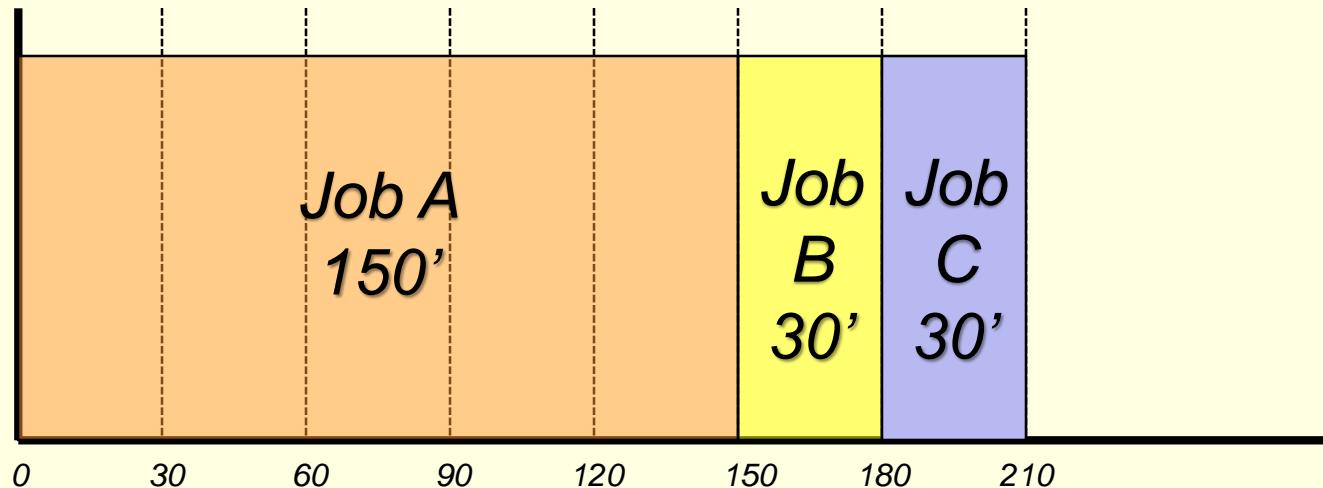


$$\text{Average Turnaround} = \left(\sum T_{\text{fim}}(i) - T_{\text{chegada}}(i) \right) / N^{\text{o}} \text{ jobs}$$

(se chegaram todos ao mesmo tempo) = $(30+60+90)/3 = 60$

Políticas de Escalonamento... (3)

□ FIFO

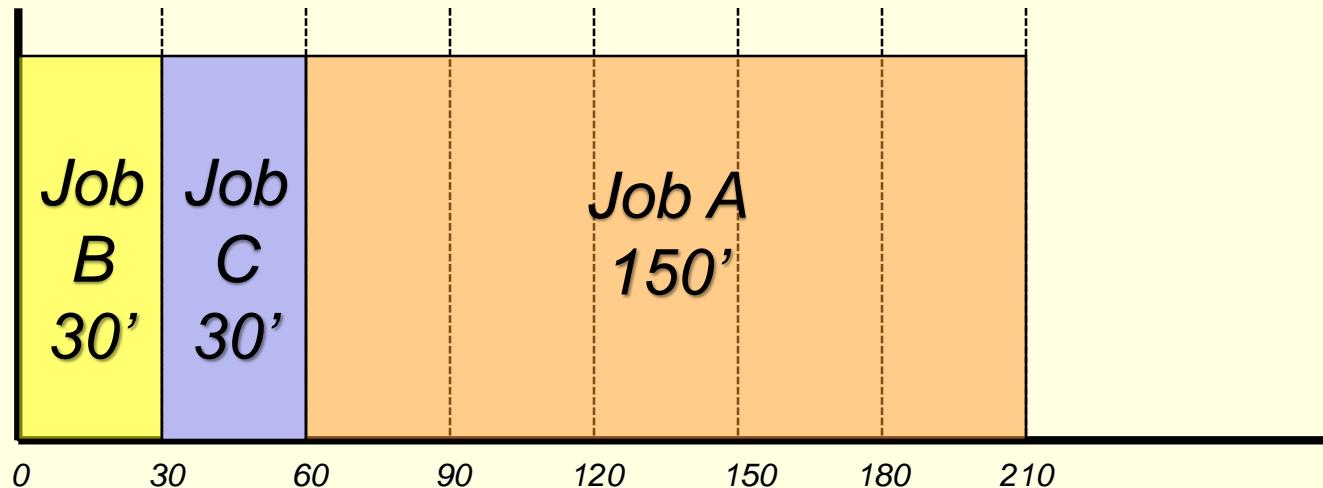


$$\text{Average Turnaround} = (\sum T_{\text{fim}}(i) - T_{\text{chegada}}(i)) / N^{\circ} \text{ jobs}$$

(se chegaram todos ao mesmo tempo) = $(150+180+210)/3 = 180$

Políticas de Escalonamento... (4)

□ Shortest Job First (SJF)

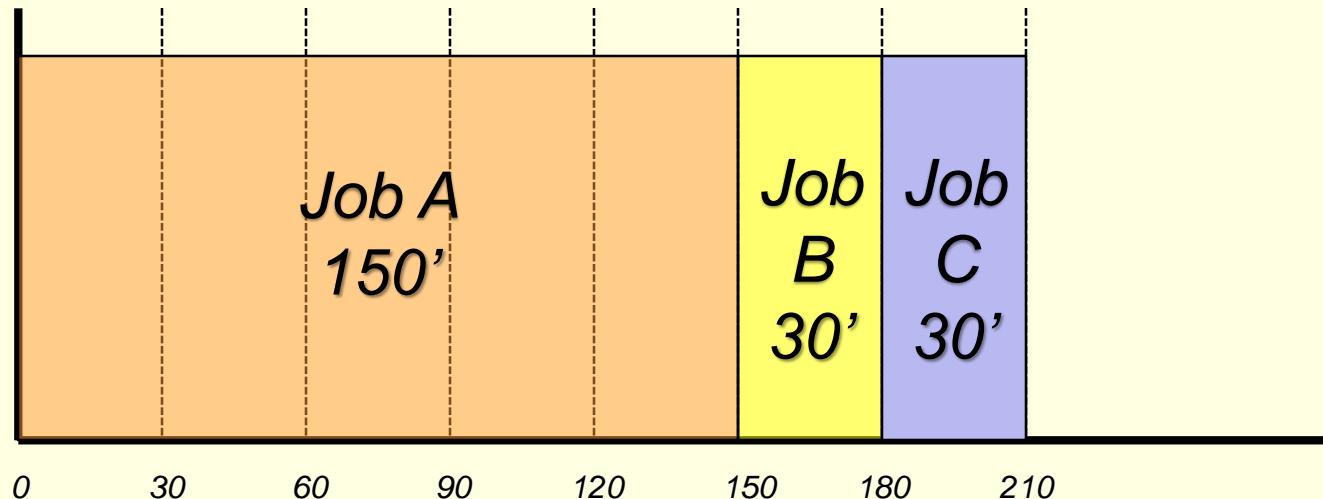


Average Turnaround = $(\sum T_{\text{fim}}(i) - T_{\text{chegada}}(i)) / N^{\circ} \text{ jobs}$

(se chegaram todos ao mesmo tempo) = $(30+60+210)/3 = 100$ reduzimos 80%

Políticas de Escalonamento... (5)

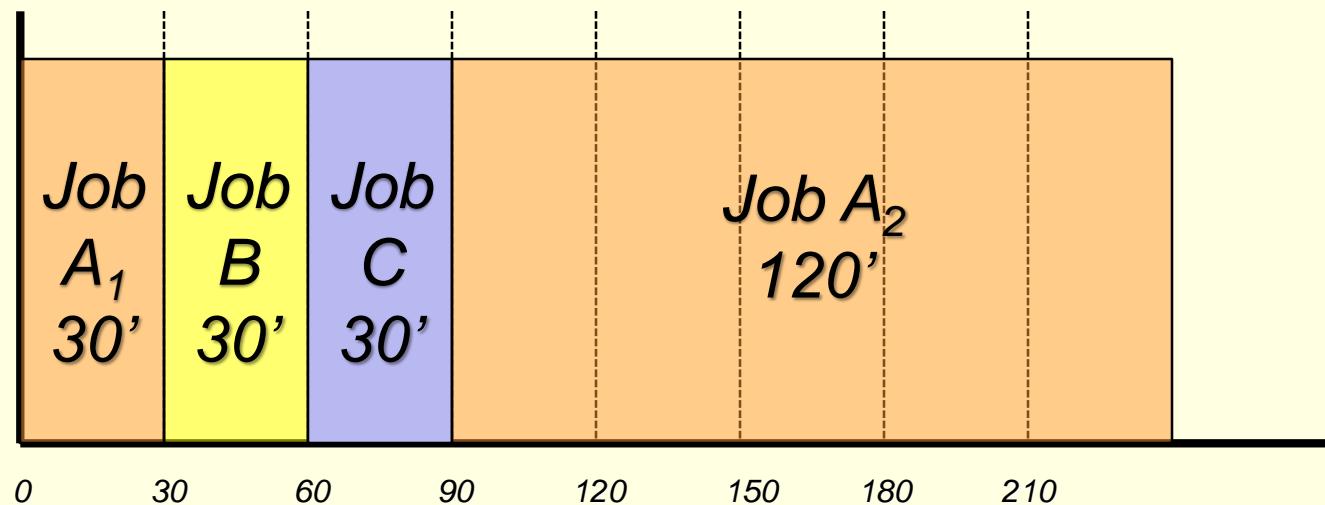
- Se B e C chegam pouco depois de A começar...
... não há nada a fazer... ou HÁ???



$$\text{Average Turnaround} = (150+180+210)/3 = 180$$

Políticas de Escalonamento... (6)

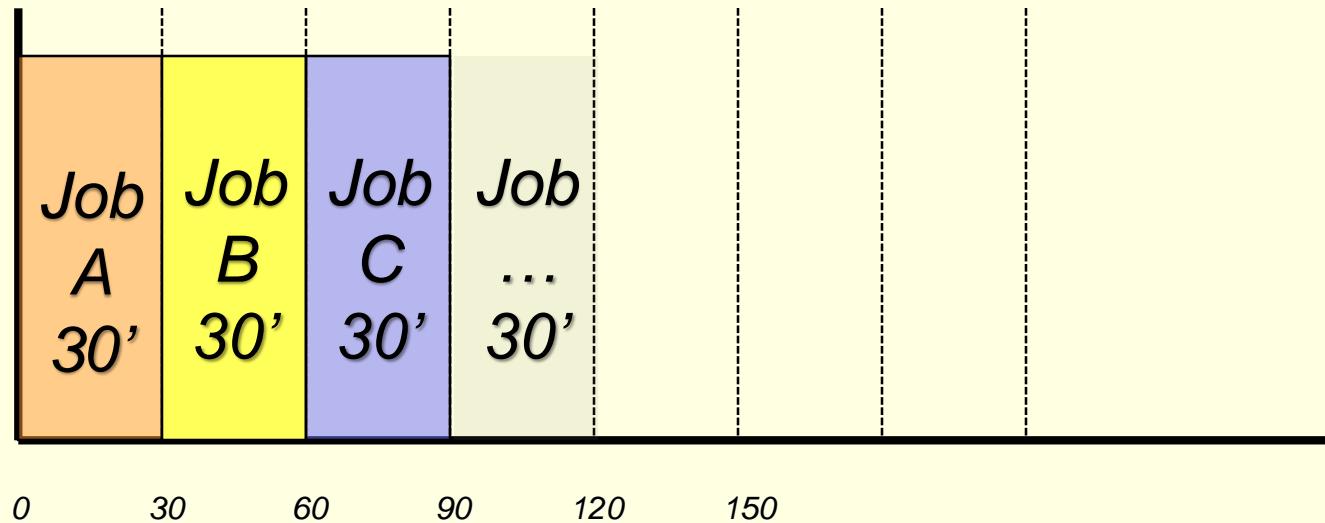
- Se B e C chegam pouco depois de A começar...
... preempção no fim do timeslot (O SO define um timeslice de activação do scheduler para que verifique se há alterações relevantes)



$$\text{Average Turnaround} = (30+60+90+210)/3 = 130 \text{ melhorou } 62\%$$

Uma outra métrica...(1)

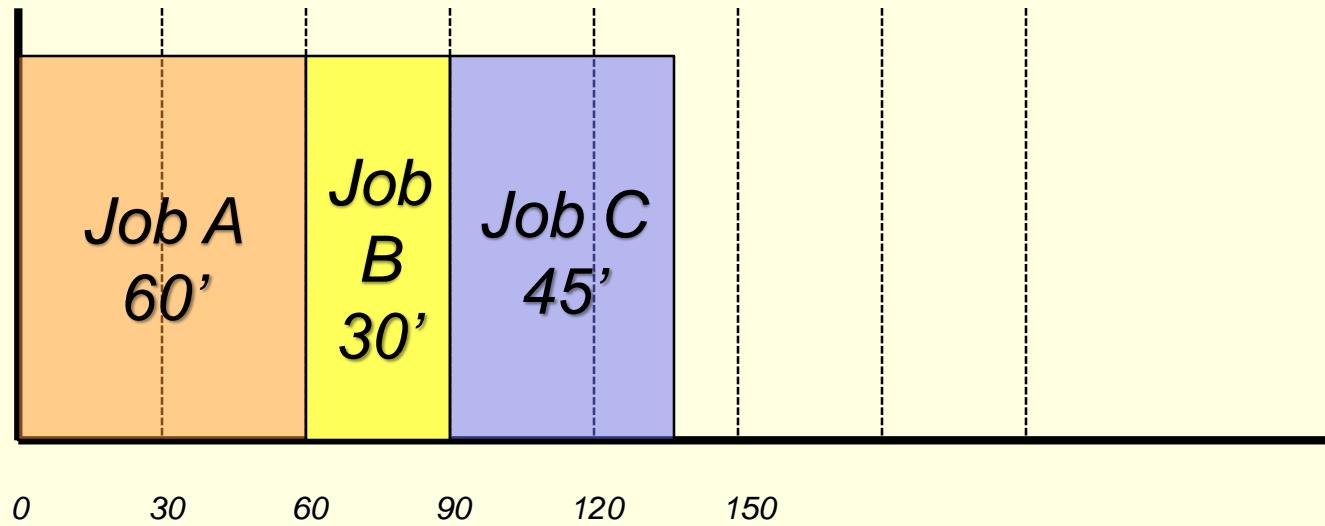
- Throughput (débito) = N° jobs / $\sum T_{\text{execução}}(i)$



$$\text{Débito} = 3/90 = 1 \text{ job a cada } 30' \text{ ou } 2 \text{ jobs/h}$$

Uma outra métrica...(2)

- Throughput= N° jobs / $\Sigma T_{\text{execução}}(i)$



$$T_{\text{put}} = 3/135 = 1 \text{ job a cada } 45' \text{ ou } 1,3 \text{ jobs/h}$$

Throughput

□ Cenário

- Às 9h da manhã, tudo em fila para entrar na loja
- No LCD está um aviso: **Average Throughput = 1.3/h**
 - Em média, as pessoas que aqui estão a ser despachadas ao ritmo de 1,3/h. Sou o 4º da fila, logo daqui a 5h12' estou despachado ☹

Fundamentos de Sistemas de Operação

Unix Windows NT Netware Mac OS DOS/V/VS Vax/VMS
Linux Solaris HP/UX AIX Mach Chorus

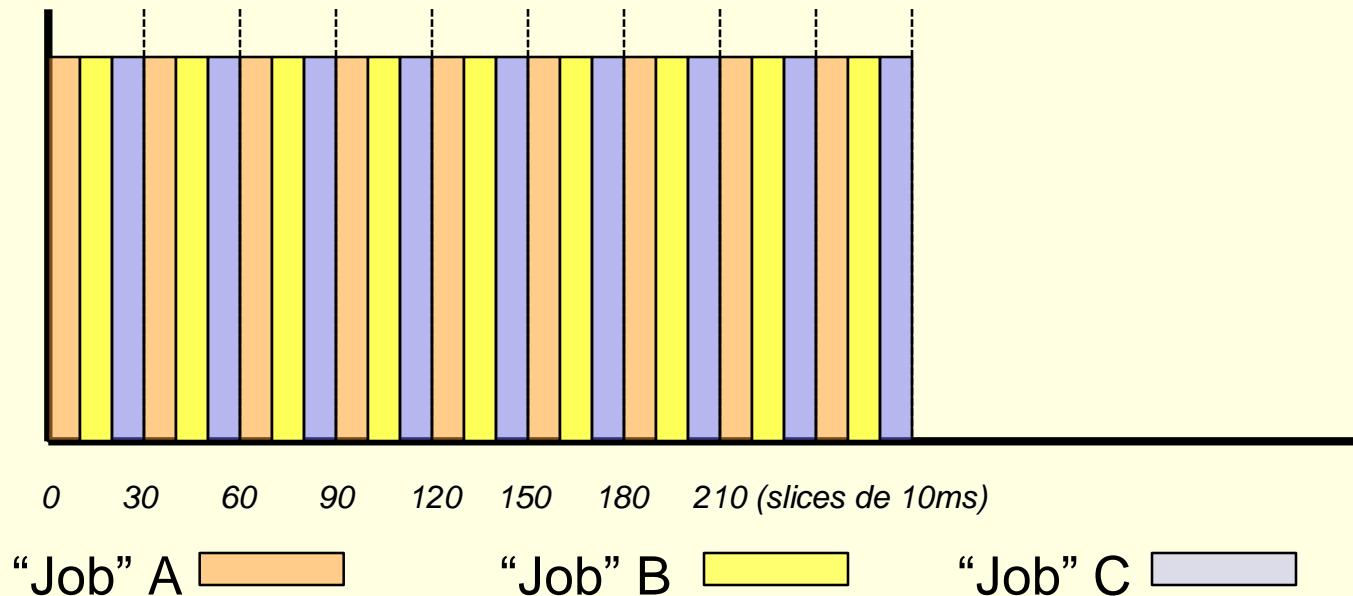
*Escalonamento do CPU:
...o início dos sistemas interactivos*

Um novo mundo...

- *Um outro tipo de sistemas...*
 - Aparecem os sistemas “*interactivos*” ou de tempo partilhado
 - Os utilizadores *interagem directamente* (por vezes à distância) com esses sistemas usando terminais, que são periféricos compostos por um monitor e um teclado
 - O escalonamento é baseado em fatia de tempo (*timeslice*), única forma de poder “atender” todos os utilizadores ☺
- ... *Uma nova métrica: Tempo de Resposta*
 - TR = tempo que medeia entre a submissão de um pedido e a resposta (e.g., desde que carregou no ENTER até ter a resposta...)
 - Embora possa ser aplicado a jobs (sem interacção), é mais “adequado” para caracterizar ambientes *interactivos*

Escalonamento Round-Robin

- $T_{\text{Resposta}} = (T_{\text{RecebeResposta}} - T_{\text{FazPedido}})$



Introdução do I/O no escalonamento

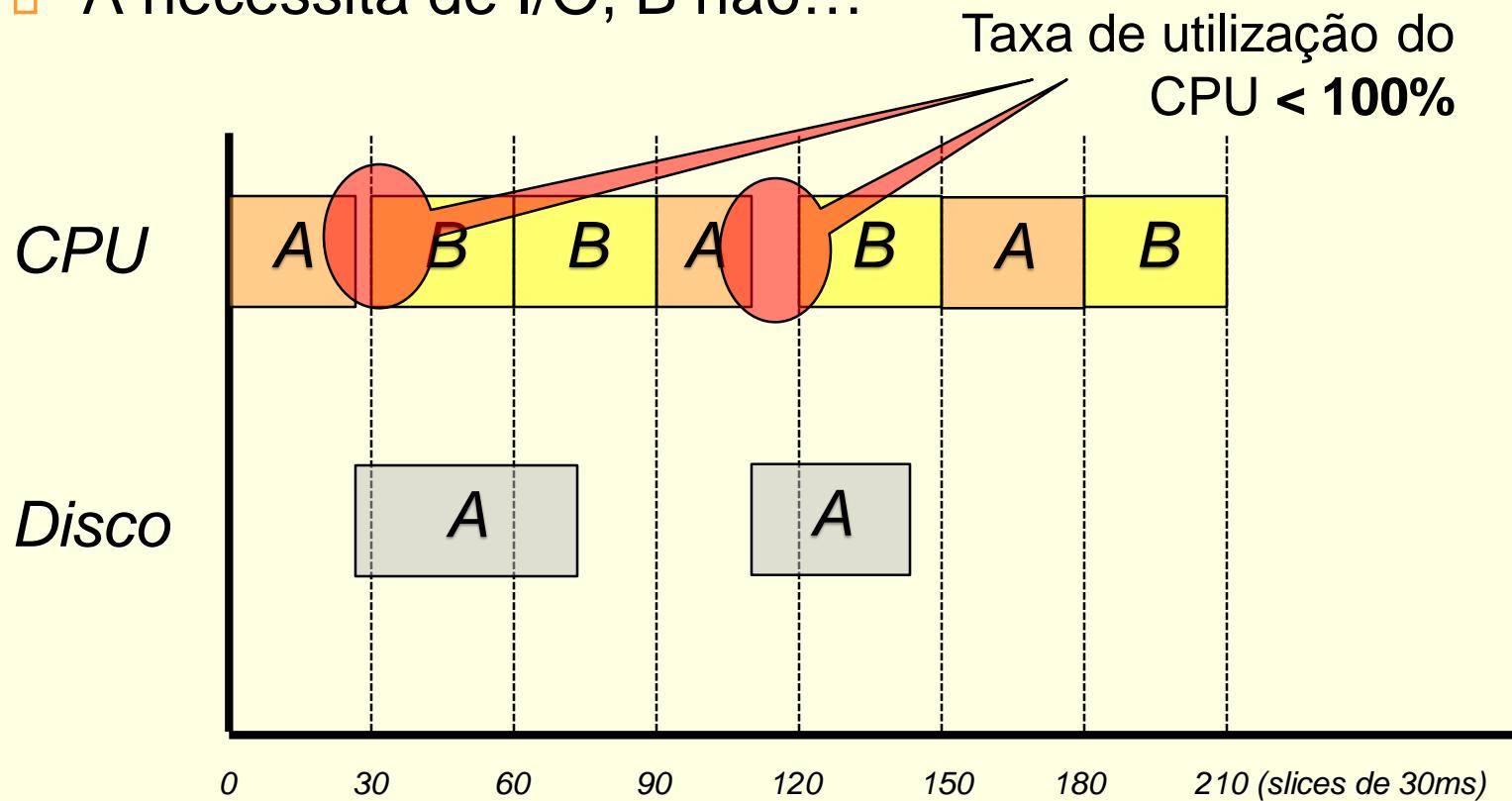
□ Como referimos anteriormente...

- *Eventos bloqueantes retiram o processo da fila ready*
 - I/O para/de periféricos (e.g., `read()` ou `write()`)
 - Instruções explícitas de bloqueio (e.g., `sleep()`)
- As operações de I/O prosseguem enquanto o CPU é dado a outro(s) processo(s)
 - Os discos magnéticos (HDD) posicionam as cabeças e transferem dados entre estas e os buffers do próprio disco
 - As unidades DMA transferem dados entre os buffers do periférico e a RAM do computador

□ Há paralelismo real entre I/O e CPU

Escalonamento considerando o I/O

- A necessita de I/O, B não...



Conclusões (1)

- *Sistemas modernos “de tipo batch”*
 - *Grandes supercomputadores (www.top500.org)*
 - O utilizador submete os programas e os dados junto com a especificação dos recursos necessários: número de nós (CPUs e RAM), tempo de execução estimado, espaço de armazenamento, etc.
 - O trabalho espera em fila até os recursos estarem disponíveis
 - É executado isoladamente nesses nós (servidores)
- *Servidores partilhados por múltiplos utilizadores*
 - Correm trabalhos diversos: servem páginas web, suportam desenvolvimento (edição/compilação/teste), armazenam dados (BDs), etc.. **Como pode um escalonador satisfazer toda a gente?**

Conclusões (2)

- Um escalonador tenta fazer “o melhor” para satisfazer requisitos (por vezes) antagónicos
 - Taxa de utilização elevada (100%)
 - Overhead baixo (decisão rápida e pouco custosa em CPU – senão está a “roubar” CPU aos utilizadores)
 - Tempo de resposta curto para processos interactivos
 - Justiça
 - Débito elevado
 - ...

Fundamentos de Sistemas de Operação

Unix Windows NT Netware Macos DOS/V/S Vax/VMS
Linux Solaris HP/UX AIX Mach Chorus

*Escalonamento do CPU:
tentando satisfazer “todo o mundo”*

Um mundo menos simples... (1)

□ Assumindo o que de facto acontece...

- Não sabemos previamente o tempo de execução de um processo
- Os processos vão chegando para serem executados e, quando em execução, executam operações de I/O e gastam CPU. Contudo, há aqueles que,
 - fazem muito mais I/O que usam CPU, em grandes volumes: **I/O bound**
 - usam muito mais CPU que I/O, em grande quantidade: **CPU bound**
 - e os que são completamente “mistos” (tipicamente **interactivos**)

Um mundo menos simples... (2)

□ Queremos um escalonador que

- Não prejudique os *interactivos* (*bom TR*) [Já reparou que por vezes (espero que não muitas) carrega nas teclas (*Word?*) e os caracteres demoram a aparecer?]
- Tenha bom débito (*despache rapidamente os processos*) e mantenha o CPU **produtivamente** ocupado)
- Não sacrifique **demasiado** os CPU-bound (*i.e.*, o favorecimento dos *interactivos* são prejudicar seriamente os outros)

O escalonador MLFQ (1)

- *Multi-Level Feedback Queue: história*
 - Trabalho de Corbato et al. em 1962, para o SO CTSS
 - Trabalho continuou no SO Multics (que inspirou o Unix)
- *MLFQ: grandes objectivos*
 - Optimizar o tempo de turnaround
 - Garantir um bom TR para os interactivos
- *Mas...*
 - **Sem saber nada sobre a duração das execuções!**

O escalonador MLFQ (2)

□ *MLFQ: princípio/conceito*

- *Ir observando a execução e ajustando as decisões; aprender com o passado para “prever” o futuro*

□ *MLFQ: estratégia*

- *N filas (ready), cada uma relativa a uma dada **prioridade** (admitamos 0 a menos prioritária e N-1 a mais prioritária)*
- *Os processos ready são dispersos pelas filas (como? a ver ☺)*
- *Quando é necessário escalar, o cabeça da fila mais prioritária é escolhido*
- *Se houver vários nessa fila, política RR entre eles*

O escalonador MLFQ (3)

□ *MLFQ: prioridades - elaborando...*

- Quando um processo novo fica ready pela 1^a vez, é colocado na fila mais prioritária...
- Mas, há medida que o tempo passa, a sua prioridade é ajustada em função do seu comportamento passado...

□ *MLFQ: formalizando um pouco (sejam A, B,... processos, P(x) a prioridade do processo x, e E(x) a execução de x)*

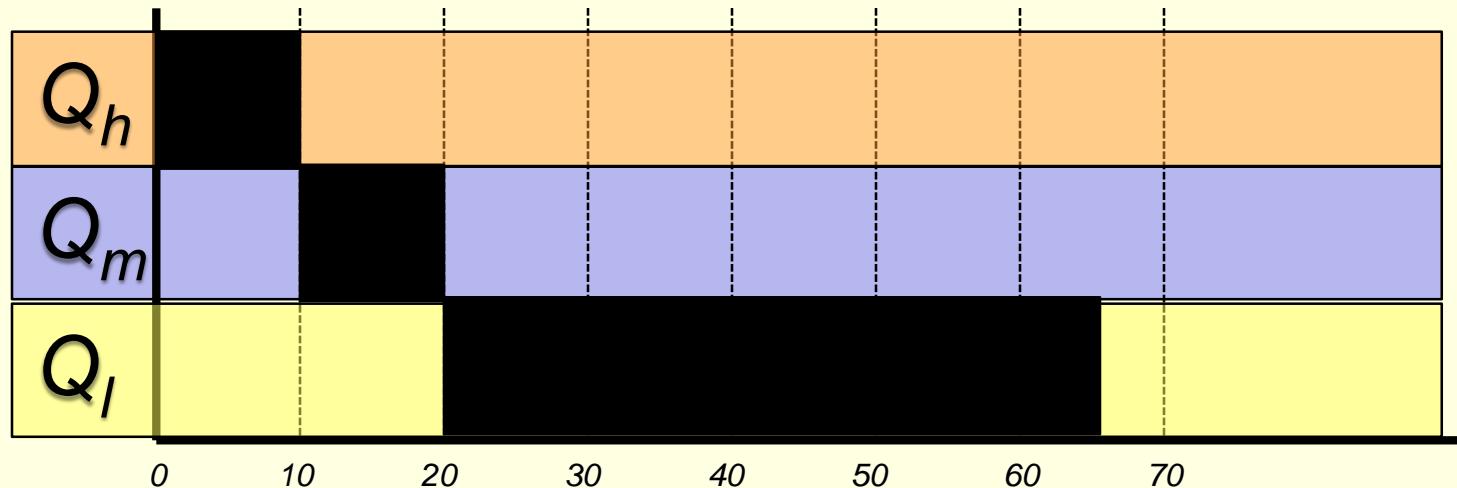
- **Regra 1:** $P(A_{novo}) = N-1$
- **Regra 2:** $P(A) > P(B) \Rightarrow E(A)$
- **Regra 3:** $P(A) = P(B) \Rightarrow$ Rodar entre $E(A), E(B)$

O escalonador MLFQ (4)

- *MLFQ: prioridades – elaborando a mudança...*
 - Se, no decurso de um timeslice, $P(x)$ não usou toda a fatia (por ter bloqueado – I/O ou outra causa) $P(x)$ mantém-se sem alteração
 - Mas se usou toda a fatia, a sua prioridade é decrementada
- *MLFQ: formalização (continuação)*
 - **Regra 4a:** Se a fatia $[T_i, T_{i+1}[$ não foi integralmente consumida, $P(x_{T_{i+1}}) = P(x_{T_i})$
 - **Regra 4b:** senão, $P(x_{T_{i+1}}) = P(x_{T_i}) - 1$

MLFQ: exemplos (1)

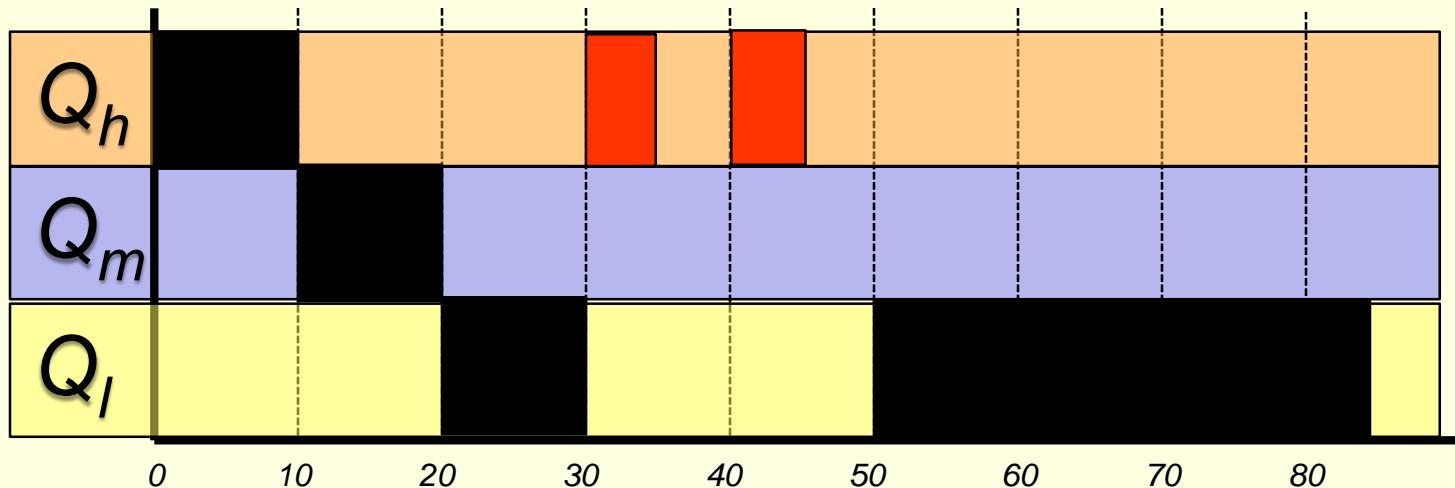
- Executando um único processo longo e CPU-bound



A execução *inicia-se com a prioridade máxima (fila Q_h)*; *finda a fatia, toda consumida, a prioridade desce (fila Q_m)*; etc...

MLFQ: exemplos (2)

- Executando um processo longo e CPU-bound (cor negra) e outro “interactivo” (vermelho)



A execucao *inicia-se com a prioridade máxima nos dois casos; contudo, enquanto o CPU-bound desce, o interactivo mantém-na...*

MLFQ: 1^a ronda de conclusões

- *Aproxima um SJF:*

- *Enquanto só existe um processo, a perda de prioridade não é relevante para a atribuição de CPU*
- *Quando “entra” o 2º processo, muito mais curto, efectivamente despacha-o depressa (mas **note-se que se fosse longo mas interactivo, o resultado era o mesmo**)*

- *Satisfaz o TR dos processos interactivos*

- *Mas... e se houver muitos interactivos???*

- *Pode nunca executar os CPU-bound... **starvation!!!***