

SO: Escalonamento

Sistemas Operacionais

2017-1

Flavio Figueiredo (<http://flaviovdf.github.io>)

Aonde Estamos

- Processos
 - Chapt 3
- Threads
 - Chapt 4
- Vamos pular o Chapt 5 brevemente
 - Sincronização e comunicação entre Processos
 - Será visto com o Chapt 7 (Deadlocks)
- [Agora] Escalonamento

Tipos de escalonadores

- Escalonadores de longo prazo
 - Balancear a carga
 - Aumentar a utilização
 - Custo de operação maior

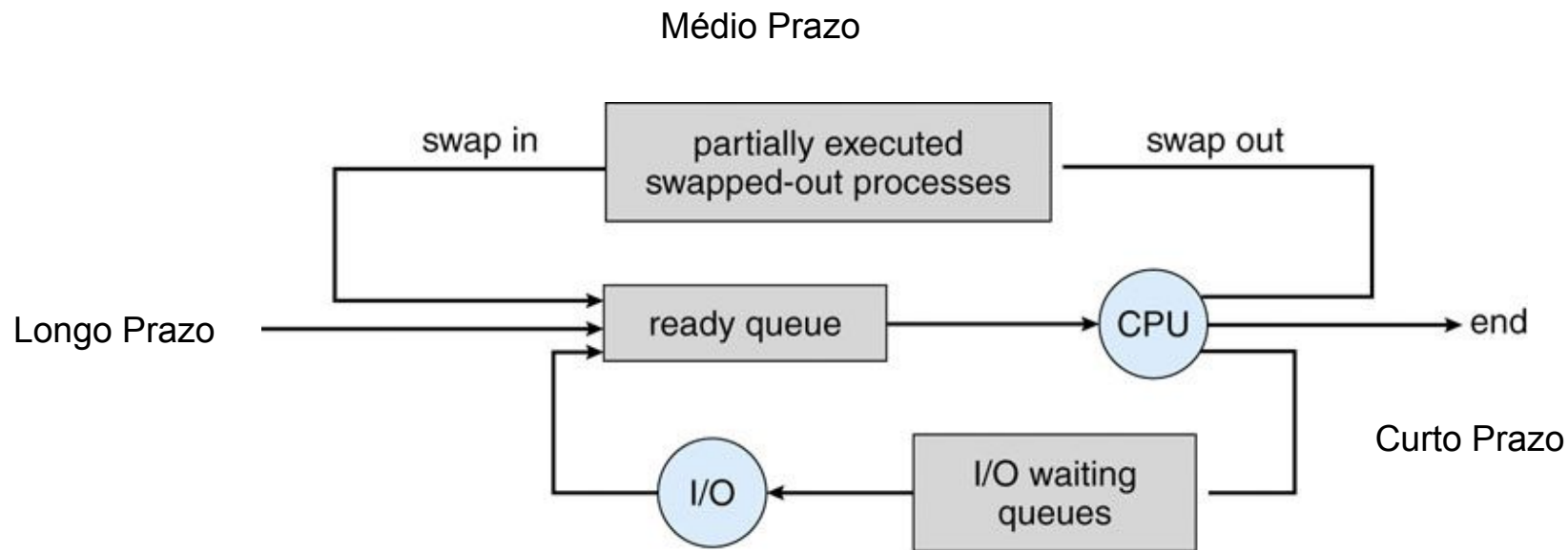
Tipos de escalonadores

- Escalonadores de longo prazo
 - Balancear a carga
 - Aumentar a utilização
 - Selecionar quais processos vão executar
 - e.g., A fila de um cluster
 - Pouco utilizado em SOs do dia-a-dia
- Escalonadores de médio prazo
 - Escolhe quais processos vão da memória para o disco
 - Processos com muita memória

Tipos de escalonadores

- Escalonadores de longo prazo
 - Balancear a carga
 - Aumentar a utilização
 - Selecionar quais processos vão executar
 - e.g., A fila de um cluster
 - Pouco utilizado em SOs do dia-a-dia
- Escalonadores de médio prazo
 - Escolhe quais processos vão da memória para o disco
 - Processos com muita memória
- Escalonadores de curto prazo
 - Como dividir o tempo entre os processos atuais
 - Nosso maior foco

Tipos de escalonadores



Como escolher qual processo executar?

- SE:
 - O SO deve terminar o maior número de tarefas possível
 - Maximizar o throughput (e.g., número de tarefas por minuto ou hora ou dia)

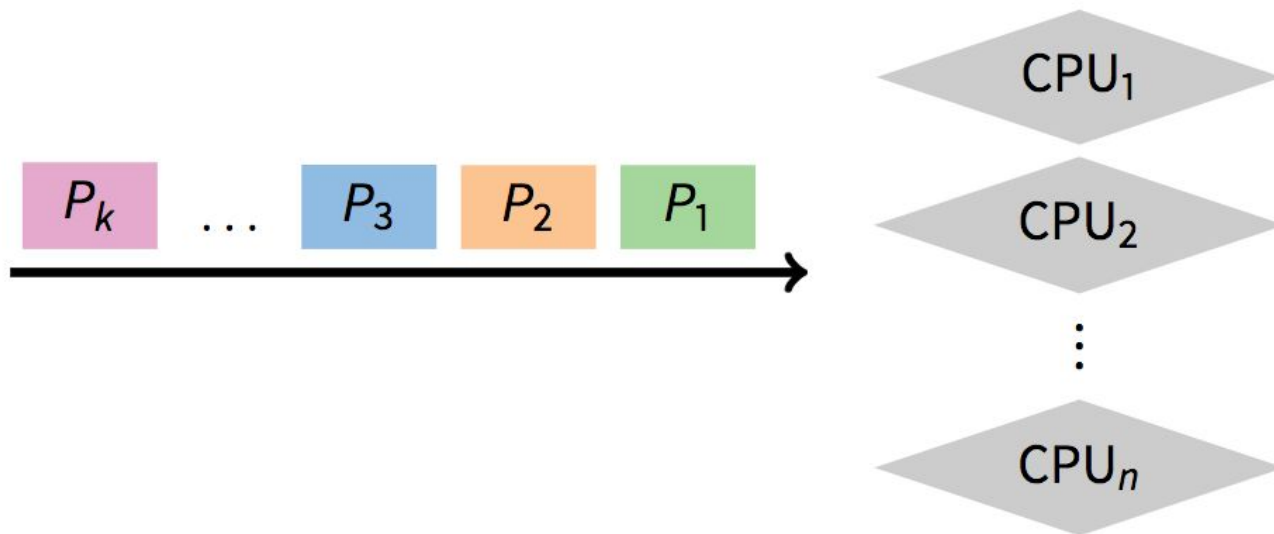
Como escolher qual processo executar?

- SE:
 - O SO deve terminar o maior número de tarefas possível
 - Maximizar o throughput (e.g., número de tarefas por minuto ou hora ou dia)
 - O SO deve garantir que cada tarefa tenha algum tempo de CPU
 - Sem starvation (passar fome)

Como escolher qual processo executar?

- SE:
 - O SO deve terminar o maior número de tarefas possível
 - Maximizar o throughput (e.g., número de tarefas por minuto ou hora ou dia)
 - O SO deve garantir que cada tarefa tenha algum tempo de CPU
 - Sem starvation (passar fome)
 - O SO deve garantir prazos
 - [Exemplo Antigo] Processar a folha de pagamento até o dia 15

O Problema de Escalonamento



- k tarefas devem executar
 - [Dinâmico] Cresce ou decresce ao longo do tempo
- $n \geq 1$ processadores
 - Não vamos nos preocupar com a diferença entre core e hyperthread

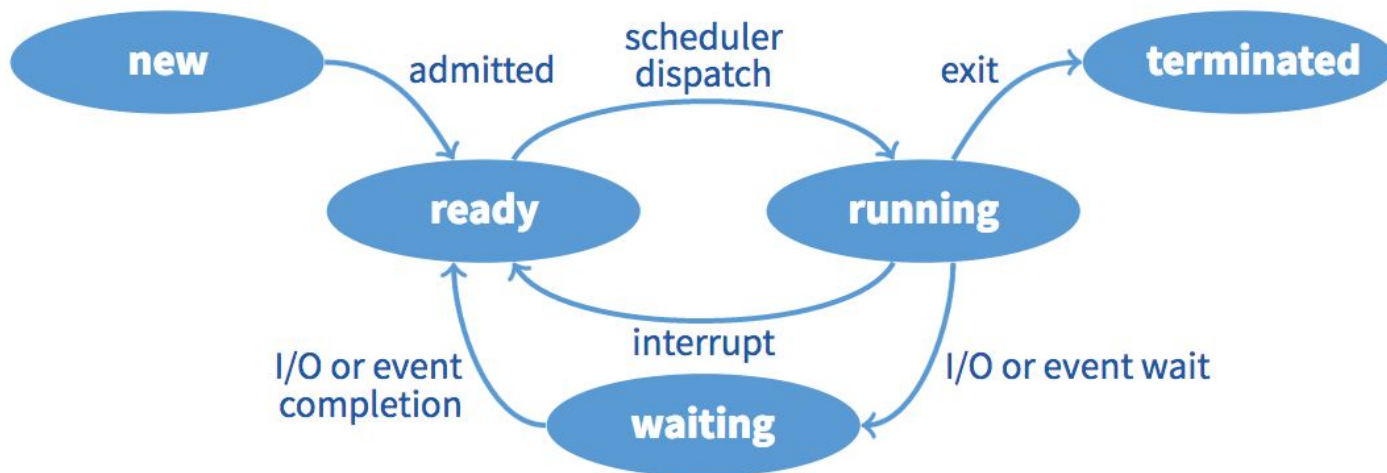
Como o sistema operacional pode contar o tempo?

Hardware Timer Interrupt

- [Lembre-se]: Sistemas operacionais são movidos a interrupções
- Hardware Timer Interrupt
 - Contar tempo
- Útil no escalonamento
 - Tal processo já executou por X interrupts
 - Cada interrupt vem em um tempo fixo
 - Cada n-ciclos da cpu

Escalonamento com e sem preempção

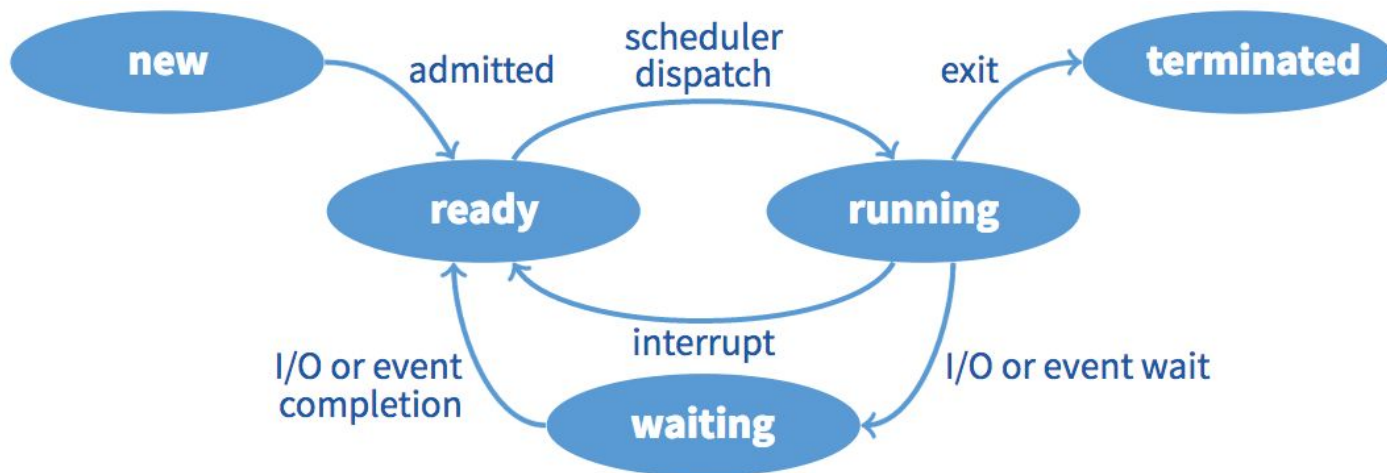
- O escalonador roda quando:
 - a. Um processo faz uma requisição de E/S (sys-call)
 - b. Quando um processo passa para o estado de *ready*
 - c. Quando um processo parra de new/waiting para *ready*
 - d. Um processo terminou



Escalonamento com e sem preempção

- O escalonador roda quando:
 - Um processo faz uma requisição de E/S (sys-call)**
 - Quando um processo passa para o estado de ready de running
 - Quando um processo passa para o estado ready de new
 - Um processo terminou**

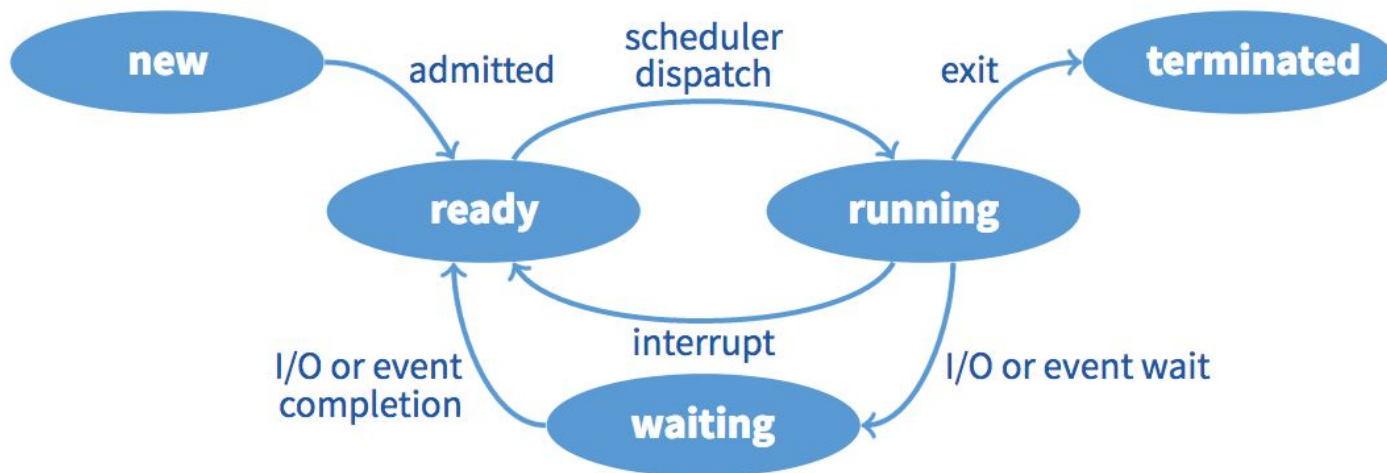
Sem Preempção
Só (a) e (d)
Qual o motivo?



Escalonamento com e sem preempção

- O escalonador roda quando:
 - a. Um processo faz uma requisição de E/S (sys-call)
 - b. Quando um processo passa para o estado de *ready*
 - c. Quando um processo para de *new/waiting* para *ready*
 - d. Um processo terminou

Com preempção



Quais as vantagens e desvantagens do escalonador preemptivo e não preemptivo?

Quais são os objetivos do escalonador?

- *Throughput (vazão)*
 - Número de processos por intervalo de tempo
 - Maior é melhor
- *Turnaround Time (TT)*
 - Tempo de término de processos (médio)
 - Menor é melhor
- *Tempo de resposta*
 - Tempo entre nascer e começar a rodar
 - Menor é menor
- Outras métricas correlatas
 - CPU Time: Tempo que a CPU fica trabalhando
 - Tempo de espera: Tempo que cada processo espera em fila

Existe um escalonamento ótimo em relação às métricas anteriores?

Considerando algumas delas ou sub-conjuntos.

First Come First Served

- Executar tarefas em ordem de chegada
 - FIFO
- Simples de implementar
 - 1 única lista simplesmente encadeada
- Throughput (número de processos por segundo)?
- Turnaround Time ou TT (tempo até cada processo finalizar)?

Processo	Tempo Exec.
P1	24s
P2	3s
P3	3s



First Come First Served

- Executar tarefas em ordem de chegada
 - FIFO
- Simples de implementar
 - 1 única lista simplesmente encadeada
- Throughput (número de processos por segundo):
 - **3 processos / 30 segundos = 0.1**
- Turnaround Time ou TT (tempo até cada processo finalizar)?
 - **(24 + 27 + 30) / 3 = 27**

Processo	Tempo Exec.
P1	24s
P2	3s
P3	3s



Mudando a ordem de chegada

- Throughput?
- TT?

Processo	Tempo Exec.
P1	24s
P2	3s
P3	3s



Mudando a ordem de chegada

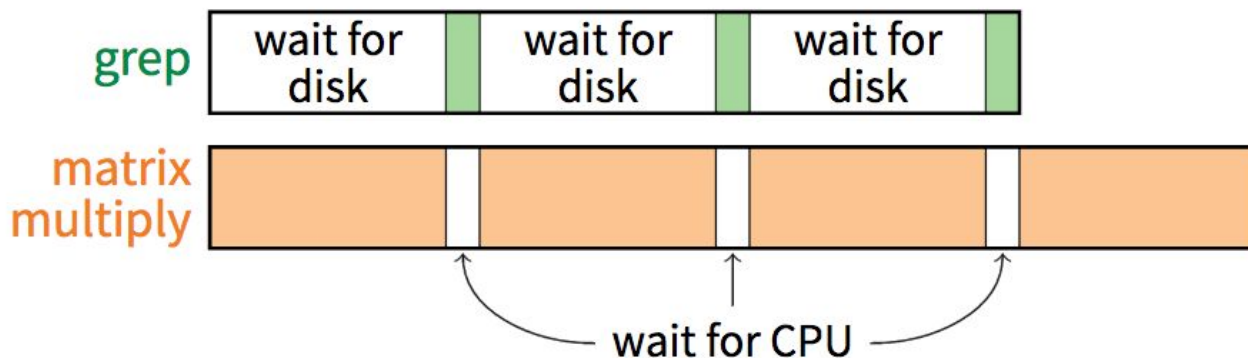
- Throughput:
 - 3 processos / 30 segundos = 0.1
- TT
 - $(30 + 3 + 6) / 3 = 13s$
 - Ganhos em TT
- [Lição 1] Escalonamento pode melhorar TT
- [Dúvida] Podemos melhorar o throughput?

Processo	Tempo Exec.
P1	24s
P2	3s
P3	3s



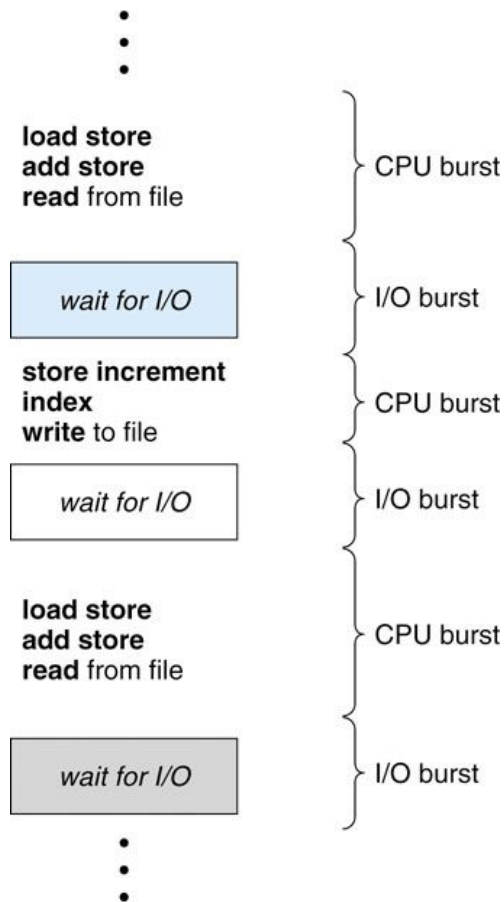
O Efeito da E/S

- Vazão pode ser melhorada por causa da E/S
- Se conseguirmos sincronizar todo E/S com o processamento de outro
 - Vazão ótima
 - Situação improvável - apenas de exemplo



Picos de CPU e picos de E/S

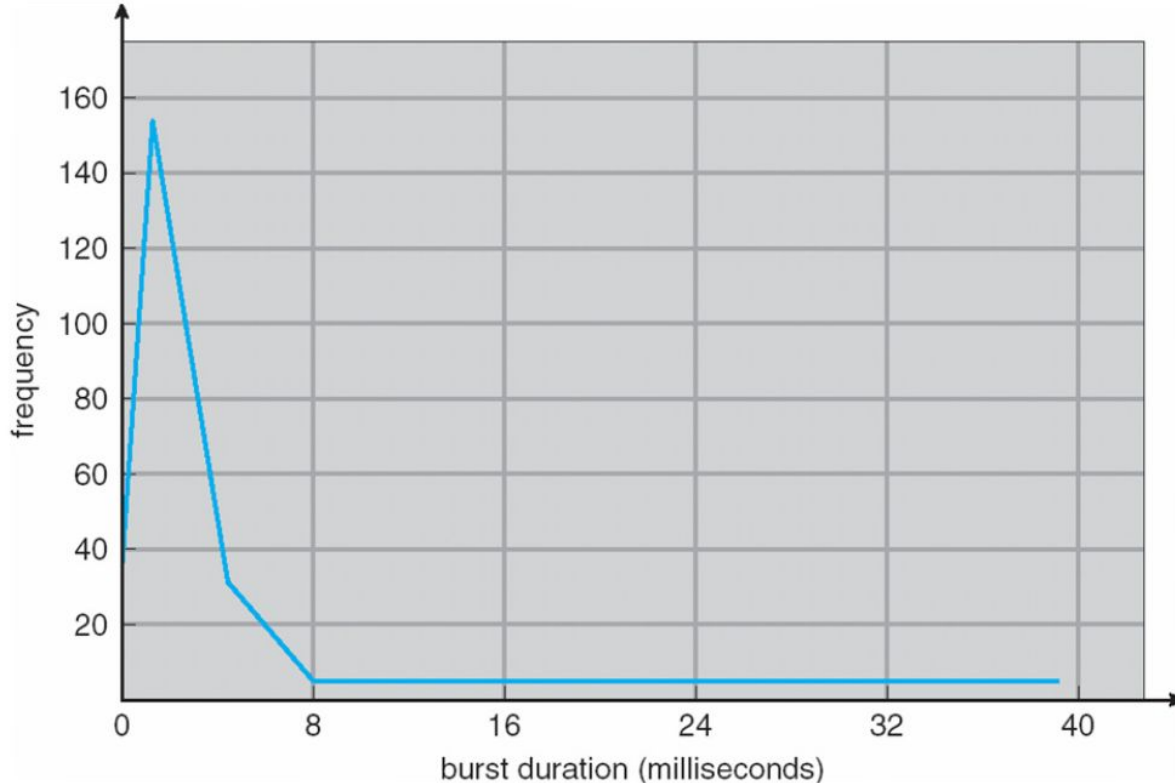
- Tempo do processo dividido entre
 - Picos de CPU
 - Picos de E/S
- Se ignorarmos *overhead* de gerenciar os dispositivos
 - Maior throughput quando maximizamos o uso concorrente de
 - CPU
 - E/S



Melhorando FCFS um pouco

- Manter a lista FIFO
- Mudar de processos nas requisições de E/S
- Quais são os problemas aqui?

Picos de CPU: Problemas para FCFS



Existem situações onde o FCFS é útil?

Shortest-job-first

- Escalone o processo com o menor pico de CPU
 - [Dúvida] Como saber o pico de CPU?
- Vamos assumir um oráculo nos primeiros slides
 - SO sabe o tempo dos processos
 - [Irreal] na maioria dos casos
 - Seria possível em um sistema onde sabemos exatamente quais tarefas serão executadas, sem interrupções e perfilamos tudo.
- Qual o objetivo do SJF?

Shortest-job-first

- Escalone o processo com o menor pico de CPU
 - [Dúvida] Como saber o pico de CPU?
- Vamos assumir um oráculo nos primeiros slides
 - SO sabe o tempo dos processos
 - [Irreal] na maioria dos casos
 - Seria possível em um sistema onde sabemos exatamente quais tarefas serão executadas.
- Qual o objetivo do SJF?
 - Minimiza o tempo de espera

SJF: Preemptivo ou Não Preemptivo

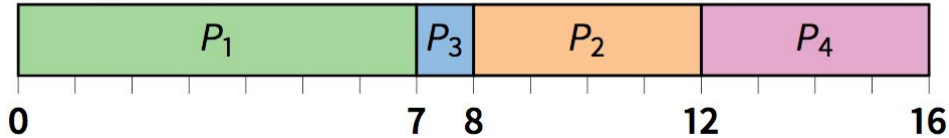
- Não preemptivo
 - A CPU é do processo até acabar

- Preemptivo
 - Caso chegue um processo de menor tempo vamos colocar o mesmo em execução

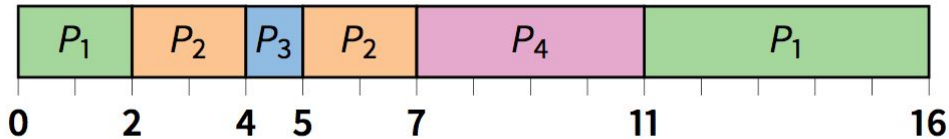
Shortest-job-first

Process	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- Non-preemptive



- Preemptive



- Drawbacks?

Quais são as vantagens e desvantagens do SJF?

SJF - Propriedades

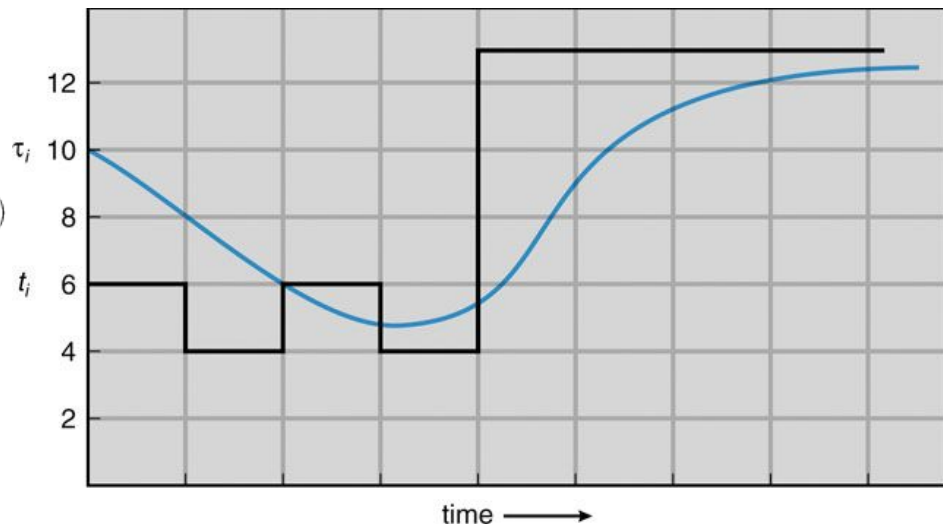
- Minimiza o tempo de espera médio
 - Os processos sempre executa o mais cedo possível
 - Todos os anteriores tem tempo menor
- É ótimo quando:
 - Não temos preempção
 - Não temos E/S
 - Menor Throughput, Turnaround Time, Tempo de espera
 - Maior uso de CPU
 - SOs modernos e de uso geral não têm tais características
 - Somos um oráculo (sabemos em quanto tempo tudo executa ou vai executar)
- Precisa estimar o tempo de pico de CPU

Estimando tempo de pico

- Smoothing exponencial

$$\begin{aligned}\tau_{n+1} &= \alpha t_n + (1 - \alpha)\tau_n \\ &= \alpha t_n + (1 - \alpha)(\alpha t_{n-1} + (1 - \alpha)\tau_{n-1}) \\ &= \alpha t_n + (1 - \alpha)(\alpha t_{n-2} + (1 - \alpha)(\alpha t_{n-2} + (1 - \alpha)\tau_{n-2}))\end{aligned}$$

- Estimar tempo de processos ainda é uma área ativa de pesquisa
 - Sistemas distribuídos
 - Sistema em nível
 - Uso de aprendizado de máquina
 - Ver conferências como USENIX
- Pouco utilizado em SOs do dia-a-dia



CPU burst (t_i)	6	4	6	4	13	13	13	...	
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

Round-Robin

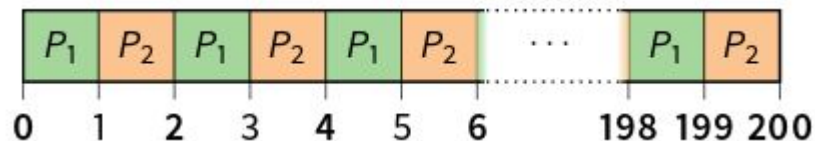
- Solução para SOs de uso geral
- Tenta minimizar:
 - Starvation (Fome)
 - Todos tem acesso a CPU
- Tenta maximizar:
 - Fairness (Igualdade)
 - Todos tem, em média, o mesmo tempo de CPU
- Não é ótimo nas métricas anterior
 - Funciona bem para nosso dia-a-dia

Round-Robin

- Cada processo executa por um tempo fixo
 - 1 Quantum
- Funciona bem com processos de tamanhos diferentes
- Abaixo
 - P1 é mais longo
 - P2: 2 quantum
 - P3: 1 quantum



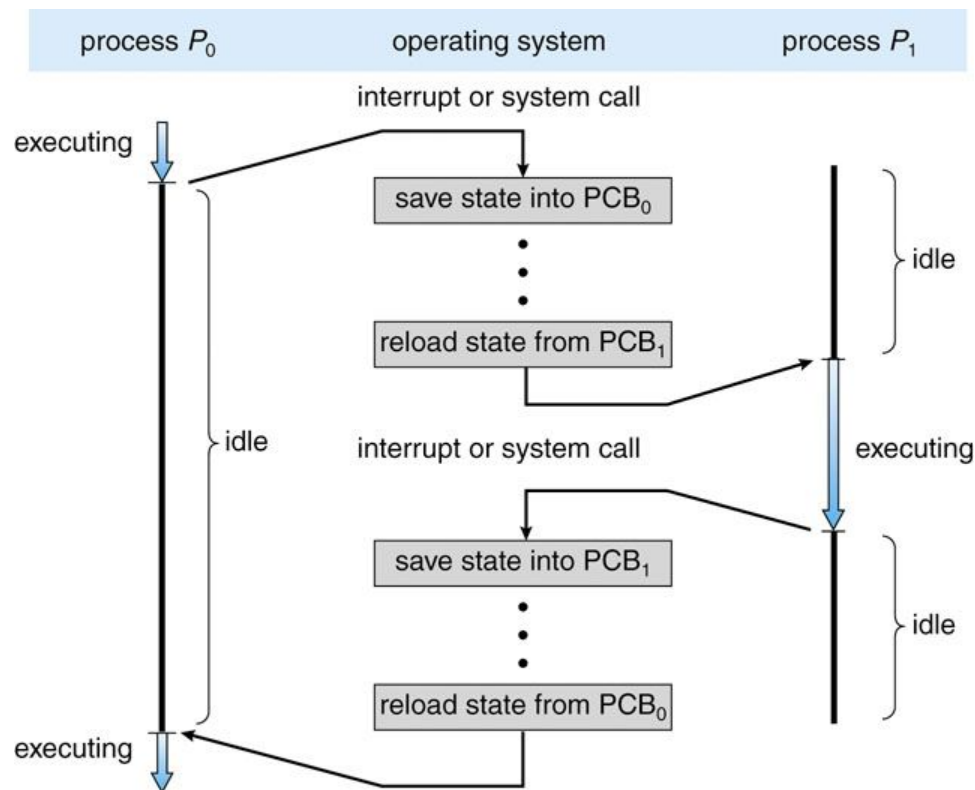
Problemas de Round-Robin



- Vamos assumir:
 - 2 Processos de 100 slots de tempo
 - 1 Quantum é 1 slot de tempo
 - Troca de contexto com pouco impacto
- Qual eficiente é o round-robin?
 - Turnaround Time
 - Throughput
 - CPU usage
- Como se compara com o FCFS?

Quais informações temos que chavear na troca de contexto?

Troca de contexto



Process Control Block



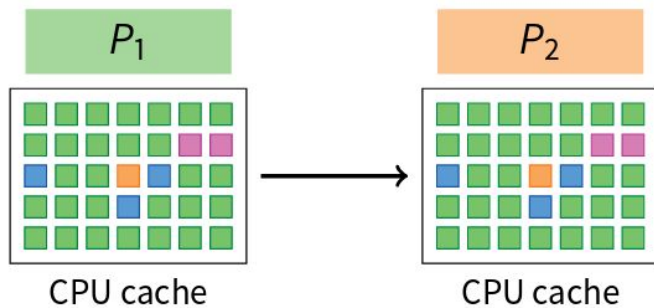
Qual o custo de troca de contexto?

Trocas de Contexto

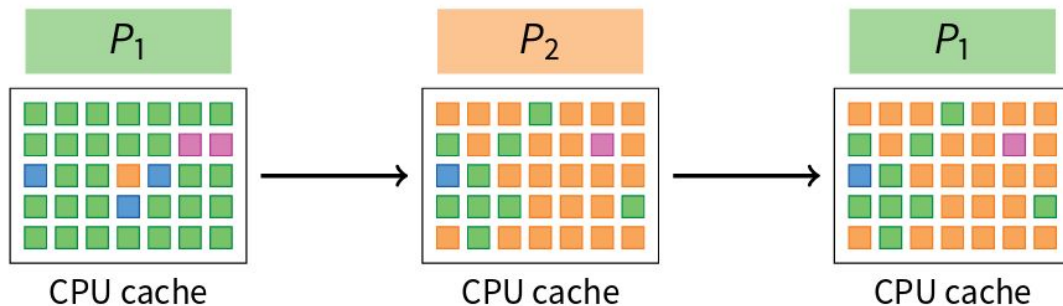
- Razoavelmente barata (pouco tempo) quando pensamos em registradores
 - Trocar espaço de endereçamento
 - Alguns registradores
 - Trocar program counter, state etc
 - Alguns registradores
- Ver XV6
 - <https://github.com/mit-pdos/xv6-public/blob/master/swtch.S>
 - <https://github.com/mit-pdos/xv6-public/blob/master/proc.h>
 - <https://github.com/mit-pdos/xv6-public/blob/master/proc.c>
 - void scheduler(void): RUNNABLE == WAITING

Custo de Cache

- Primeira Troca

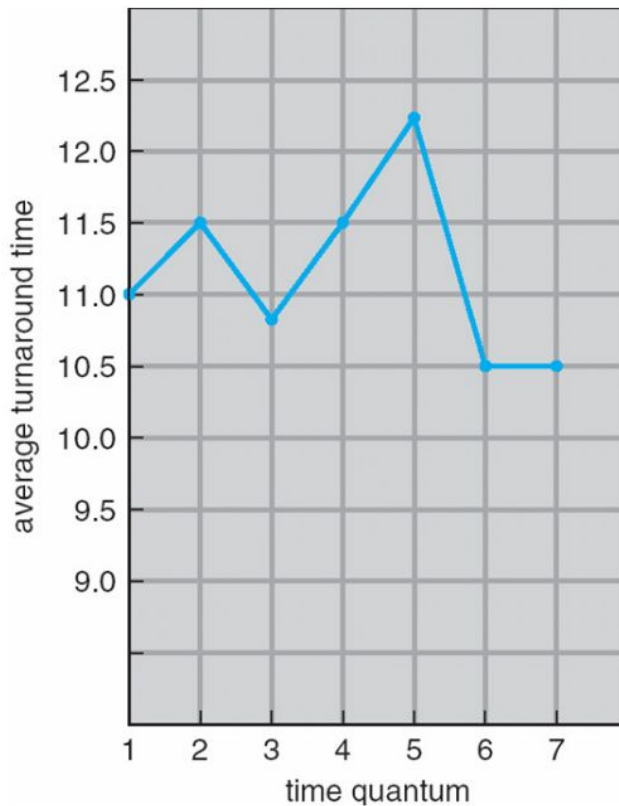


- Segunda Troca



Escolhendo o Quantum

- Muito Pequeno
 - Trocas de contexto podem degradar o sistema
- Muito grande
 - Starvation
 - Infinito = FIFO
- Geralmente:
 - 1-100 ms



process	time
P_1	6
P_2	3
P_3	1
P_4	7

Médio Prazo

- Partição SWAP
- Memória Virtual no Windows
- Alto custo de troca de contexto
 - Disco é bem mais lento do que memória
- Quantums maiores
- Foco em alguns processos
 - Dependendo do tamanho
 - Se memória estiver OK, podemos não precisar

Para onde vamos...

- [Mais detalhes] Escalonamento com Prioridade
 - Já passamos rapidamente
- [Mais detalhes] Escalonamento de Threads
 - Já passamos rapidamente
- Escalonamento em CPUs multicore
- Escalonador $O(1)$ do Linux

- No livro:
 - Silbershchatz: Chapt 6
 - XV6 Book: Chapt 1
 - Tanenbaum: Chapt 2.4