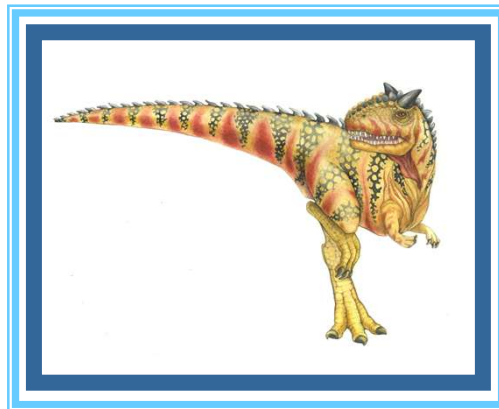


# Capítulo 5: Escalonamento da CPU



# Sobre a apresentação (About the slides)



Os slides e figuras dessa apresentação foram criados por Silberschatz, Galvin e Gagne em 2009. Essa apresentação foi modificada por Cristiano Costa (cac@unisinós.br). Basicamente, os slides originais foram traduzidos para o Português do Brasil.

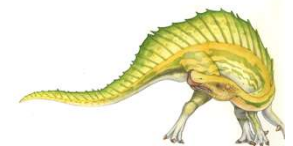
É possível acessar os slides originais em <http://www.os-book.com>  
Essa versão pode ser obtida em <http://www.inf.unisinós.br/~cac>



The slides and figures in this presentation are copyright Silberschatz, Galvin and Gagne, 2009. This presentation has been modified by Cristiano Costa (cac@unisinós.br). Basically it was translated to Brazilian Portuguese.

You can access the original slides at <http://www.os-book.com>

This version could be downloaded at <http://www.inf.unisinós.br/~cac>

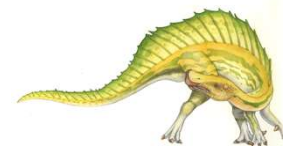




# Capítulo 5: Escalonamento de CPU

---

- Conceitos Básicos
- Critérios de Escalonamento
- Algoritmos de Escalonamento
- Escalonamento de Threads
- Escalonamento de Multiprocessadores
- Exemplos de Sistemas Operacionais
- Avaliação de Algoritmos

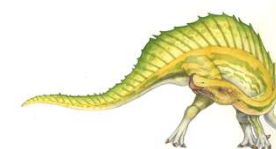


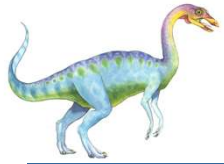


# Objetivos

---

- Introduzir escalonamento de CPU, que é a base para sistemas operacionais multiprogramados
- Descrever vários algoritmos de escalonamento de CPU
- Discutir critérios de avaliação para selecionar algoritmos de escalonamento de CPU para um sistema em particular

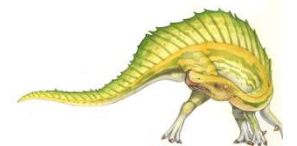




# Conceitos Básicos

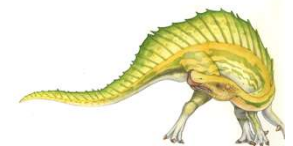
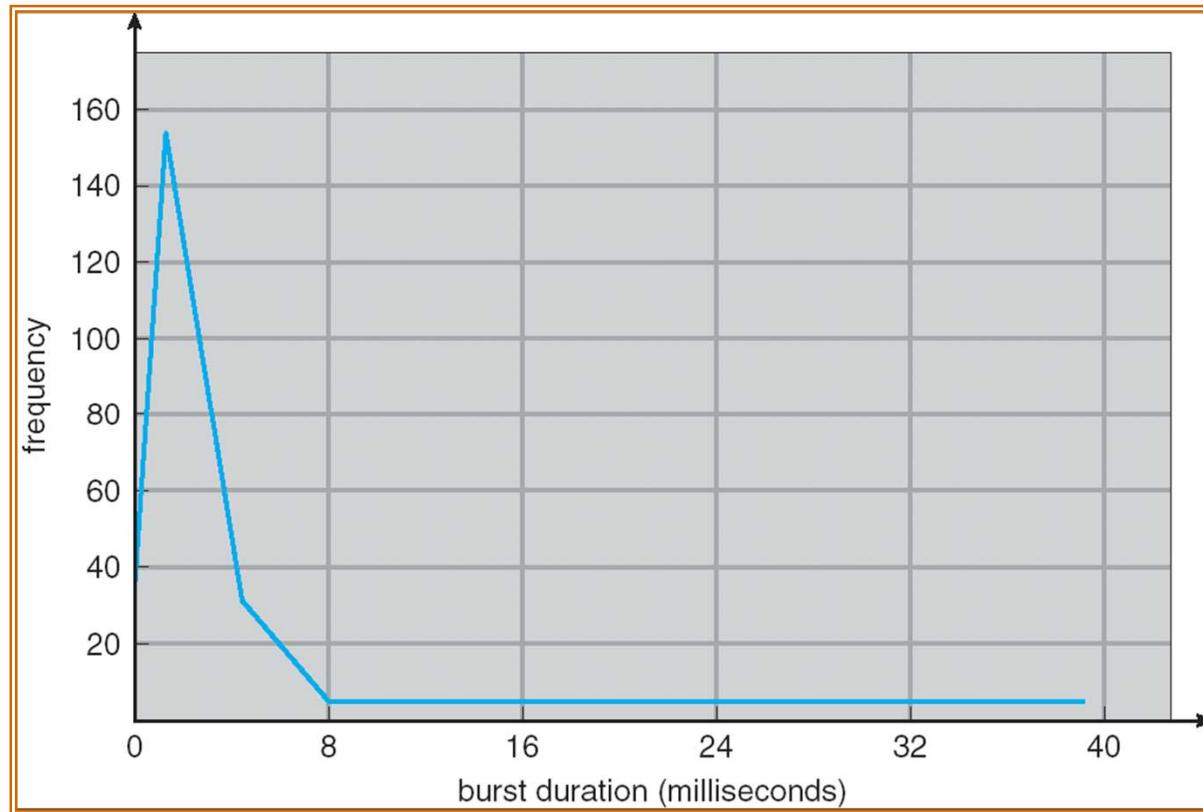
---

- Máxima utilização da CPU é obtida com multiprogramação
- Fase de uso da CPU e de E/S – Execução de processos consiste de uma fase de execução da CPU e espera por E/S
- Distribuição das fases de uso da CPU (*CPU bursts*)



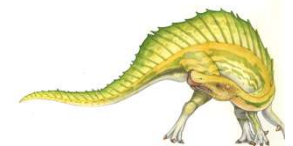
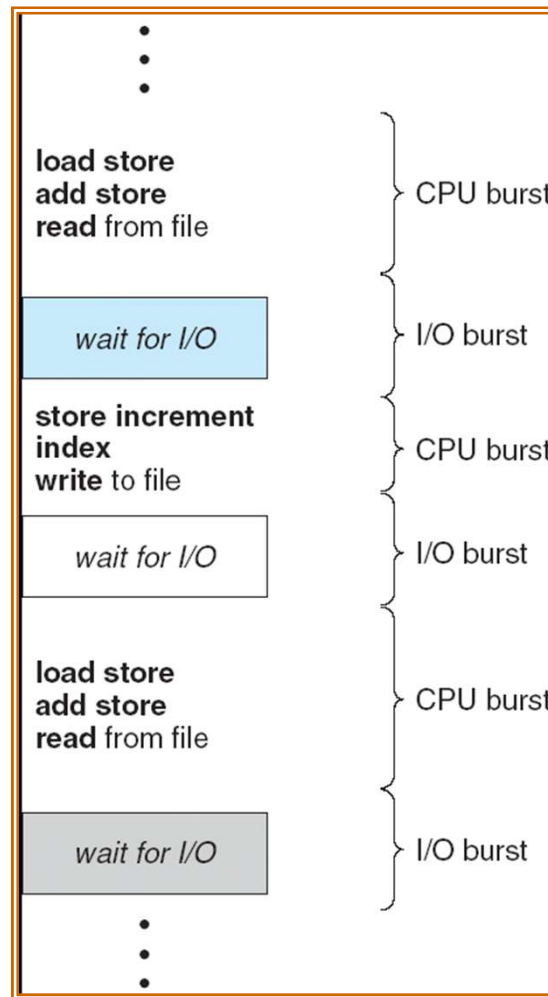


# Histograma de duração de fases de uso da CPU





# Alternando fases de uso da CPU e E/S

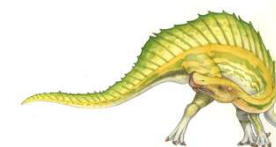




# Escalonador da CPU

---

- Seleciona dentre os processos na memória em estado pronto, e aloca a CPU para um deles.
  
- Decisões de escalonamento da CPU ocorrem quando um processo:
  1. Muda do estado *executando* para *esperando*.
  2. Muda do estado *executando* para *pronto*.
  3. Muda do estado *esperando* para *pronto*.
  4. Termina.
  
- Escalonamento nas condições 1 e 4 é *não-preemptivo*.
  
- Todas as outras alocações são *preemptivas*.



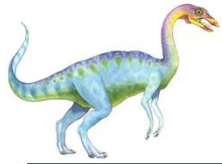


# Despachante

---

- O despachante (*dispatcher*) é o módulo que fornece o controle da CPU ao processo selecionado pelo escalonador da CPU. Essa função envolve:
  - Troca de contexto
  - Mudança para o modo usuário
  - Desvio para o endereço adequado no programa do usuário, para reiniciar o programa
  
- *Latência de Despacho* – tempo gasto pelo despachante para interromper a execução de um processo e iniciar a execução de outro.

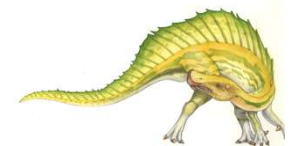


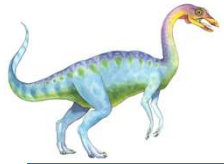


# Critérios de Alocação

---

- **Utilização da CPU** – manter a CPU ocupada a maior parte possível do tempo
- **Produtividade** (*Throughput*) – número de processos que completam sua execução por unidade de tempo
- **Tempo de Processamento** (*Turnaround*) – quantidade de tempo necessária para executar um determinado processo
- **Tempo de Espera**– quantidade de tempo que um processo esteve esperando na fila de processos prontos
- **Tempo de Resposta** – intervalo de tempo entre o envio de uma requisição e a produção da primeira resposta, não a saída (para ambientes tempo compartilhado)

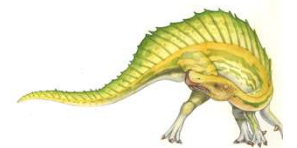




# Critérios de Otimização

---

- Maximizar utilização da CPU
- Maximizar produtividade
- Minimizar o tempo de processamento
- Minimizar o tempo de espera
- Minimizar o tempo de resposta

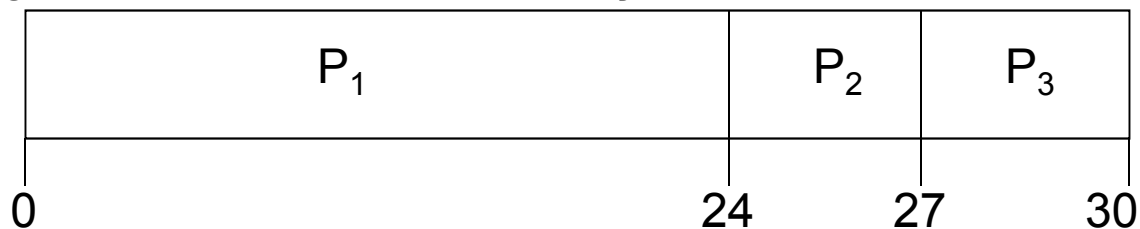




# Primeiro a Chegar, Primeiro a ser Servido (FCFS\*)

<u>Processo</u>	<u>Tempo de Rajada</u>
$P_1$	24
$P_2$	3
$P_3$	3

- Suponha que os processos chegam na ordem:  $P_1$ ,  $P_2$ ,  $P_3$   
O diagrama de Gantt para a alocação é::



- Tempo de espera para  $P_1 = 0$ ;  $P_2 = 24$ ;  $P_3 = 27$
- Tempo de espera médio:  $(0 + 24 + 27)/3 = 17$

\*FCFS - *First Come, First Served*



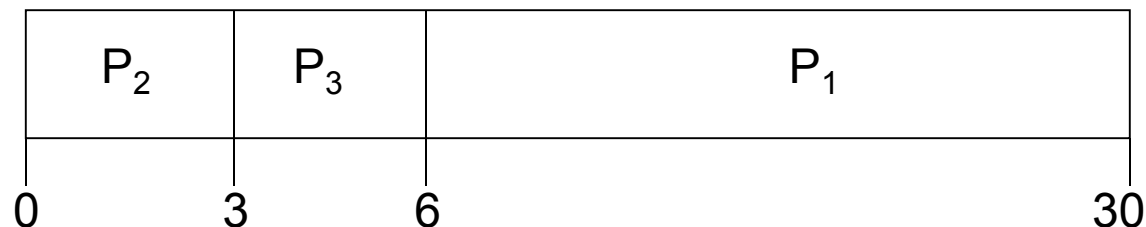


## Escalonamento FCFS (Cont.)

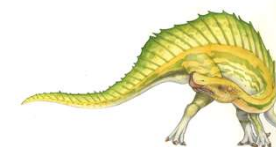
Suponha que os processos cheguem na ordem

$$P_2, P_3, P_1$$

- O diagrama de Gantt para a alocação é:



- Tempo de espera para  $P_1 = 6$ ;  $P_2 = 0$ ;  $P_3 = 3$
- Tempo de espera médio :  $(6 + 0 + 3)/3 = 3$
- Bem melhor que o caso anterior.
- *Efeito Comboio*: processos curtos após processos longos



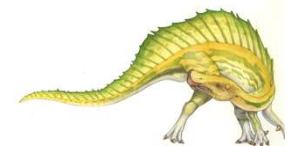


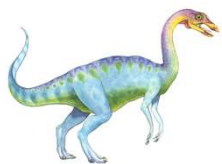
# Escalonamento Menor Job Primeiro (SJR\*)

---

- Associado com cada processo a duração da sua próxima fase de uso da CPU. Uso dessas durações para escalonar o processo com o menor tempo.
- SJF é ótimo— permite o menor tempo médio de espera para um dado conjunto de processos.
  - A dificuldade é saber o tamanho da próxima requisição de CPU

\*SJR - *Shortest-Job-First*

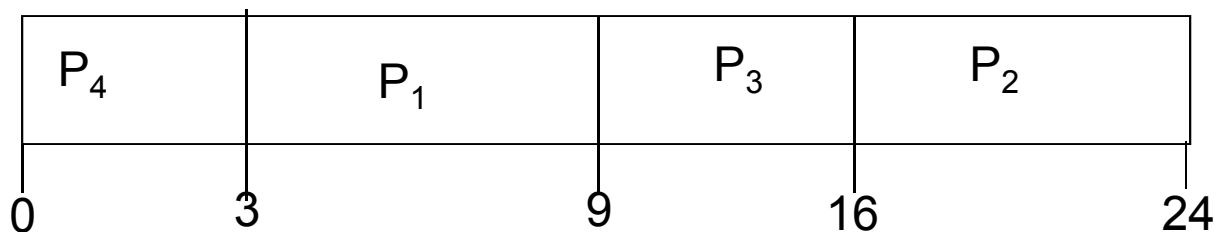




# Exemplo de SJF

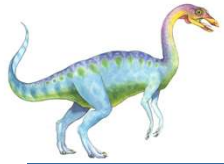
<u>Processos</u>	<u>Tempo de Rajada</u>
$P_1$	6
$P_2$	8
$P_3$	7
$P_4$	3

- Diagrama de escalonamento SJF:



- Tempo de espera médio =  $(3 + 16 + 9 + 0) / 4 = 7$



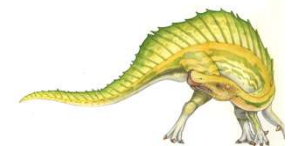


# Determinando a duração da próxima fase de uso da CPU

- Pode somente determinar a duração.
- Pode ser calculada com a duração de uso da CPU na fase anterior, usando médias exponenciais.

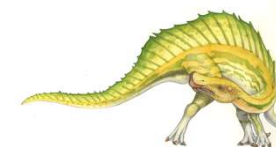
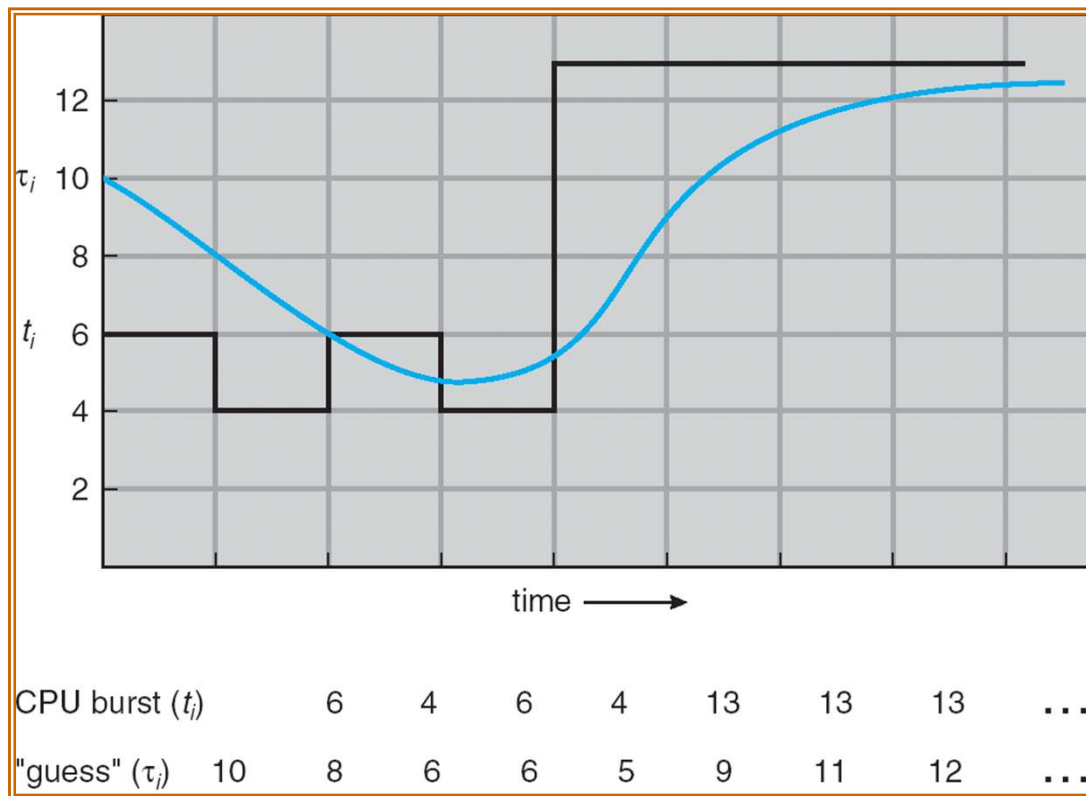
1.  $t_n$  = duração real da enésima fase de uso da CPU
2.  $\tau_{n+1}$  = valor previsto para próxima fase de uso da CPU
3.  $\alpha, 0 \leq \alpha \leq 1$
4. Defina :

$$\tau_{n+1} = \alpha t_n + (1 - \alpha)\tau_n.$$





# Previsão da Duração da Próxima Fase de Uso da CPU





# Exemplos de média exponencial

- $\alpha = 0$

- $\tau_{n+1} = \tau_n$
- História recente não é levada em consideração.

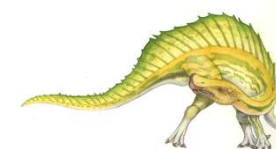
- $\alpha = 1$

- $\tau_{n+1} = t_n$
- Somente a duração da fase de uso da CPU mais recente conta.

- Se a fórmula for expandida, temos:

$$\begin{aligned}\tau_{n+1} = & \alpha t_n + (1 - \alpha) \alpha t_{n-1} + \dots \\ & + (1 - \alpha)^j \alpha t_{n-j} + \dots \\ & + (1 - \alpha)^{n-1} t_0\end{aligned}$$

- Uma vez que tanto  $\alpha$  quanto  $(1 - \alpha)$  são menores ou iguais a 1, cada termo sucessivo tem peso menor que o seu predecessor.





# Escalonamento por Prioridade

- Um número de prioridade (inteiro) é associado com cada processo
- A CPU é alocada para o processo com a maior prioridade (menor inteiro  $\equiv$  maior prioridade).
  - Preemptivo
  - Não-preemptivo
- SJF é um escalonador com prioridade, no qual a prioridade é a previsão da próxima fase de uso da CPU.
- Problema  $\equiv$  *Starvation* (abandono de processo) – processos de baixa prioridade podem nunca executar.
- Solução  $\equiv$  *Aging* (envelhecimento) – ao passar do tempo, aumentar a prioridade dos processos que ficam em espera no sistema.

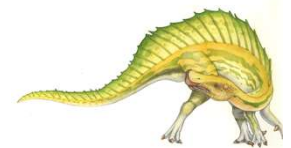




# Alocação Circular - Round Robin (RR)

---

- Cada processo recebe uma pequena unidade de tempo de CPU (*quantum*), usualmente 10-100 milissegundos. Depois de transcorrido este tempo, o processo é preemptado e adicionado ao fim da fila de processos prontos.
  
- Se existem  $n$  processos na fila de processos prontos e o *quantum* é  $q$ , então cada processo obtém  $1/n$  do tempo da CPU em blocos de no máximo  $q$  unidades de tempo de uma vez. Nenhum processo espera mais do que  $(n-1)q$  unidades de tempo.
  
- Desempenho
  - $q$  alto  $\Rightarrow$  FIFO
  - $q$  pequeno  $\Rightarrow q$  deve ser maior que a troca de contexto, senão a sobrecarga é muito alta.

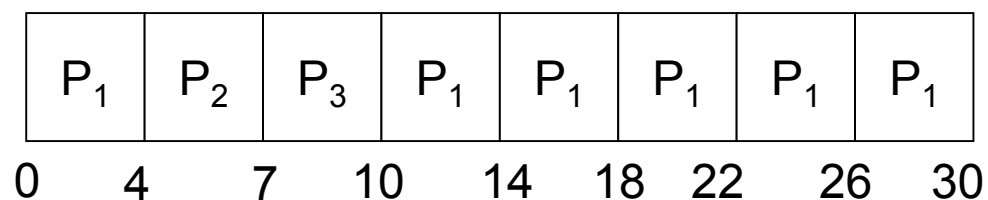




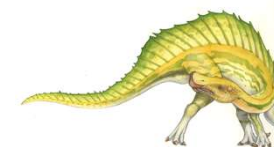
## Exemplo: RR com quantum= 4

<u>Processo</u>	<u>Tempo de Rajada</u>
$P_1$	24
$P_2$	3
$P_3$	3

- O diagrama de Gantt é:



- Tipicamente, possui maior tempo médio de processamento do que o SJF, mas melhor *resposta*.

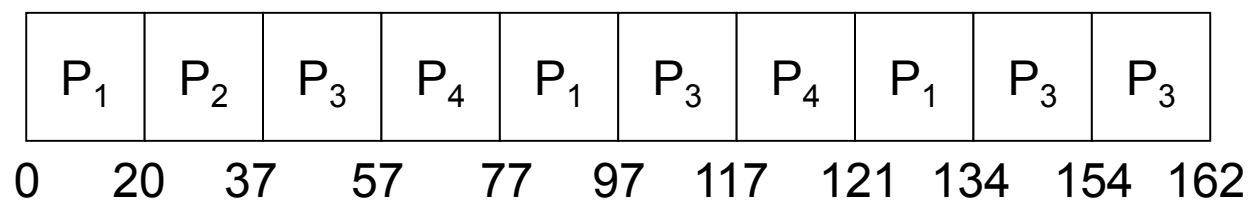




## Exemplo: RR com quantum= 20

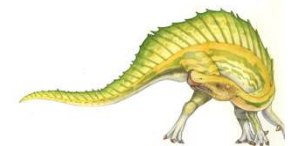
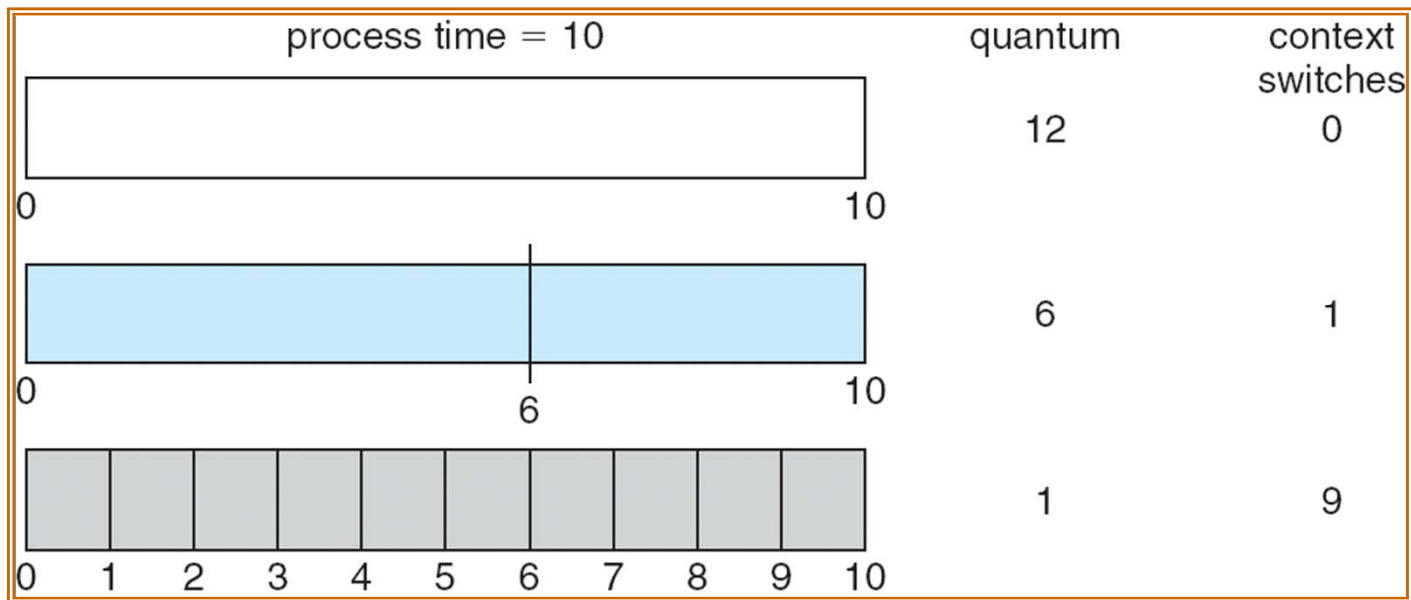
<u>Processo</u>	<u>Tempo de Rajada</u>
$P_1$	53
$P_2$	17
$P_3$	68
$P_4$	24

- O diagrama de Gantt é:



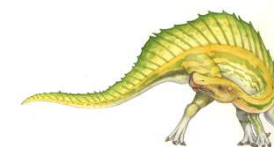
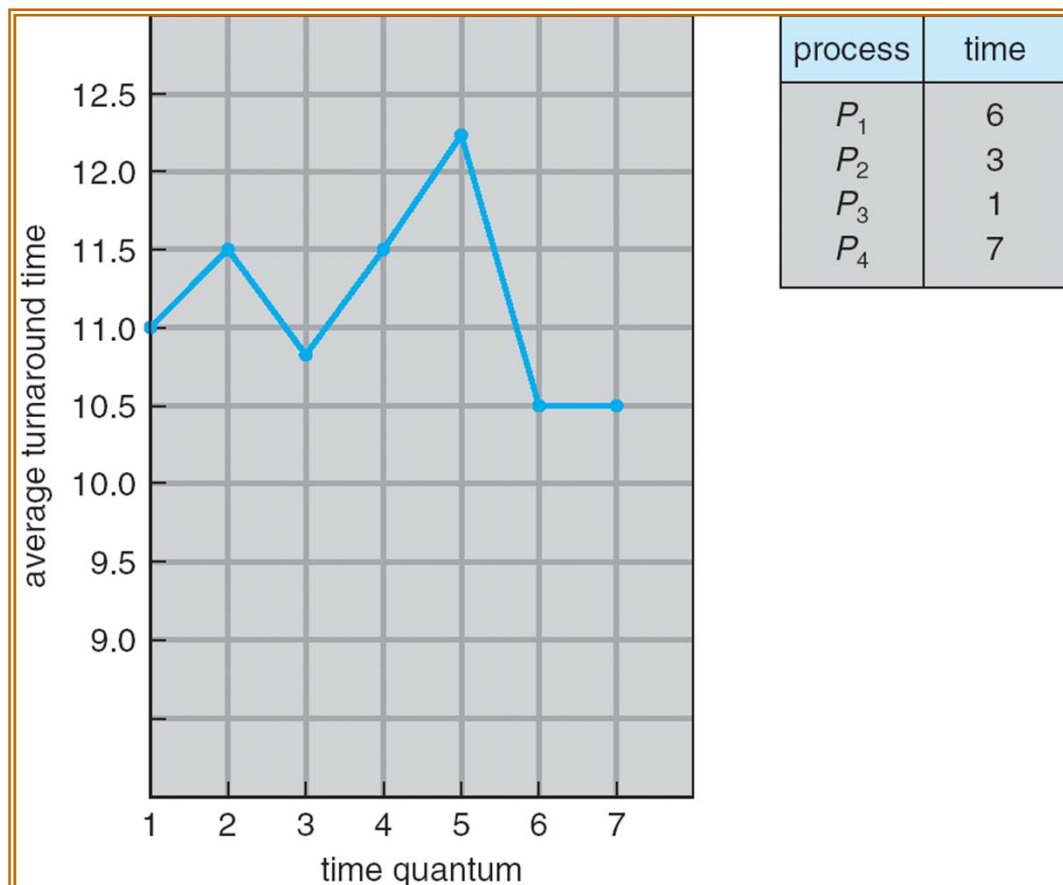


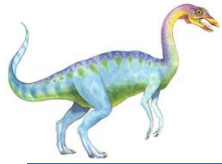
# O Quantum e o Tempo de Troca de Contexto





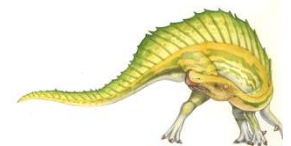
# Variação do tempo de processamento com o quantum





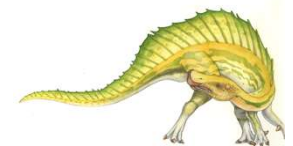
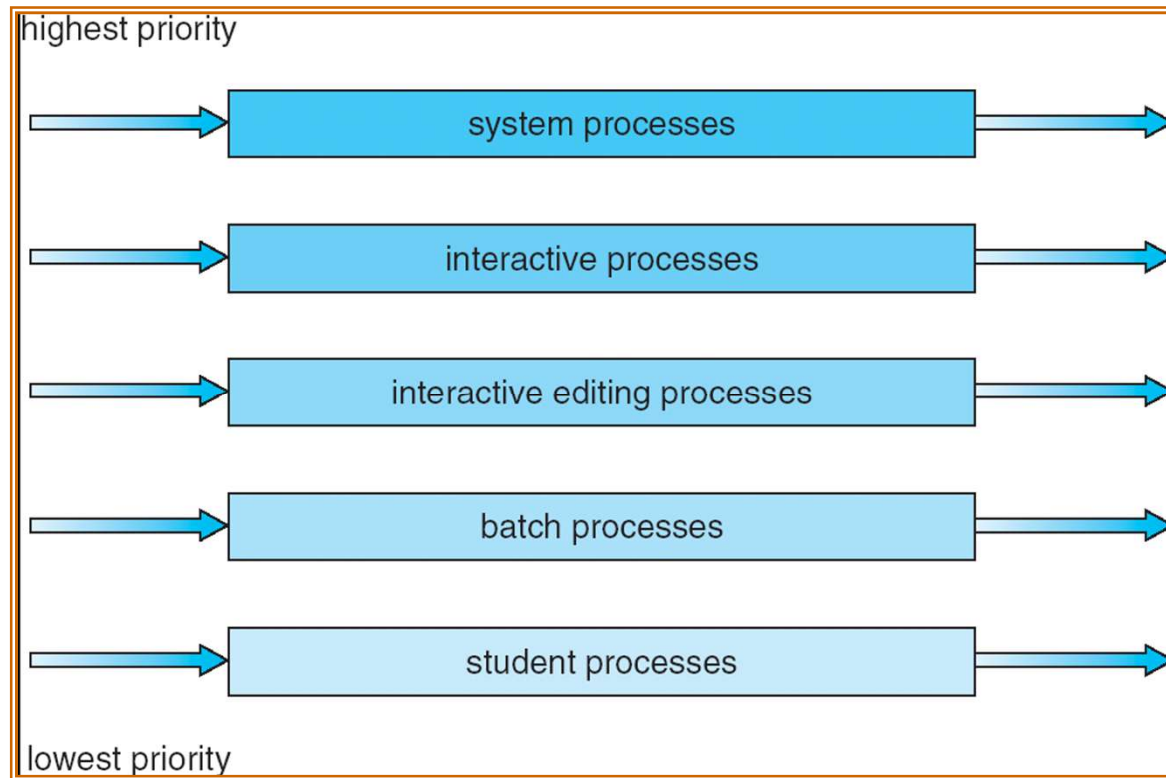
# Alocação com Múltiplas Filas

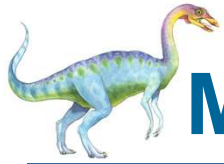
- Fila de processos prontos é particionada em filas separadas:  
primeiro plano - *foreground* (interativo)  
segundo plano - *background* (*batch*)
  
- Cada fila tem seu próprio algoritmo de escalonamento,  
interativo – Alocação Circular (RR)  
batch – Primeiro a chegar, Primeiro a ser servido (FCFS)
  
- Escalonamento deve ser realizado entre as filas.
  - Escalonamento de prioridade fixa; Ex.: a fila de processos interativos pode ter prioridade absoluto sobre a fila de processos batch. Possibilidade de *starvation*.
  - Fatia de tempo – cada fila recebe certa quantia de tempo de CPU que poderia ser então alocada aos processos dessa fila; Ex.:  
80% para processos interativos em RR  
20% para processos *batch* em FCFS





# Escalonamento com Múltiplas filas

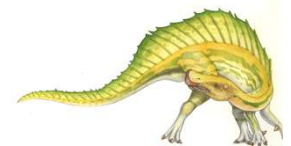




# Múltiplas Filas com Realimentação

---

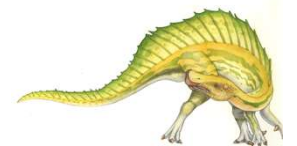
- Um processo pode se mover entre as várias filas; envelhecimento (*aging*) pode ser implementado desta forma.
  
- Escalonamento com múltiplas filas e transferências entre as filas é definido pelos seguintes parâmetros, a saber:
  - Número de filas
  - Algoritmos de escalonamento para cada fila
  - Método usado para determinar quando transferir um processo para uma fila de prioridade mais alta
  - Método usado para determinar quando transferir um processo para uma fila de prioridade mais baixa
  - Método usado para determinar em qual fila um processo deve ser colocado, quando precisar usar a CPU

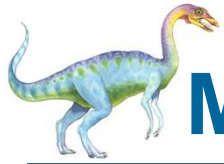




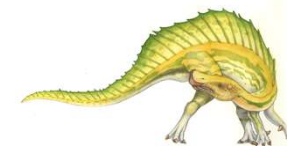
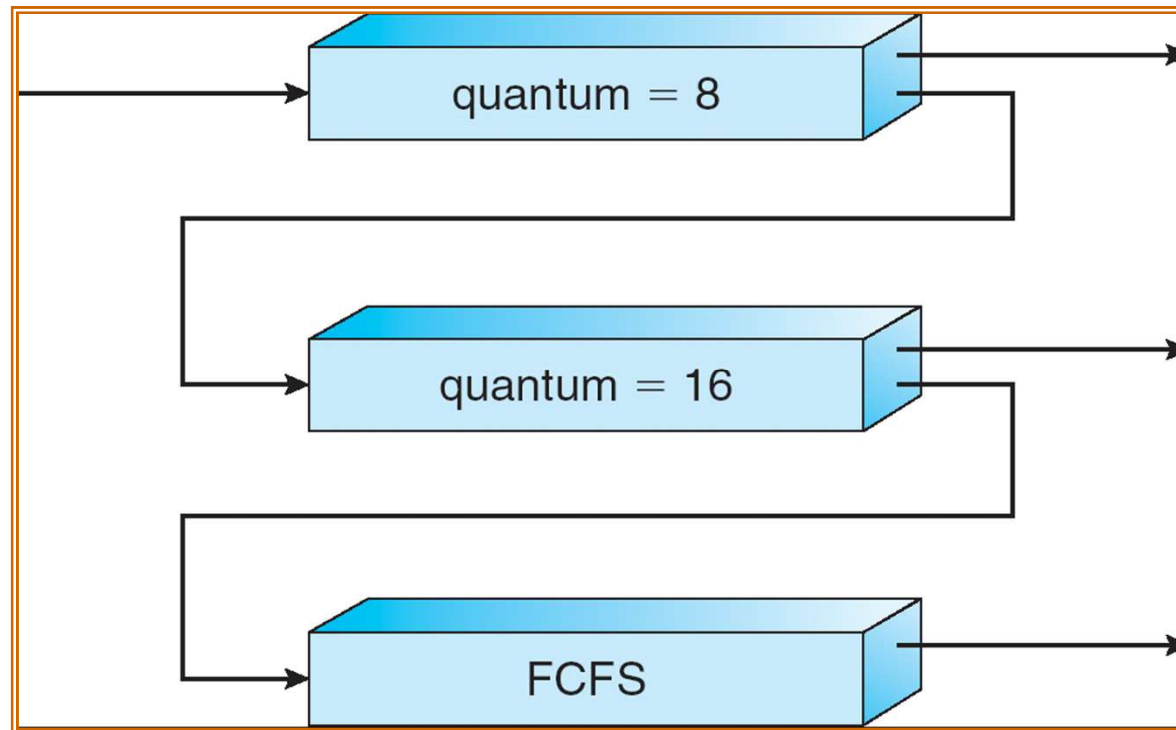
# Exemplo de Múltiplas Filas com Realimentação

- Três filas:
  - $Q_0$  – RR com quantum de 8 milissegundos
  - $Q_1$  – RR com quantum 16 milissegundos
  - $Q_2$  – FCFS
  
- Escalonamento
  - Um novo job entra na fila  $Q_0$  a qual utiliza RR. Quando ele ganha a CPU, job recebe 8 milissegundos. Se ele não finalizar em 8 milissegundos, o job é movido para a fila  $Q_1$ .
  - Na fila  $Q_1$  o job é de novo servido por RR e recebe 16 milissegundos adicionais. Se ele ainda não completou, é preemptado e movido para a fila  $Q_2$ .





# Múltiplas Filas com Realimentação

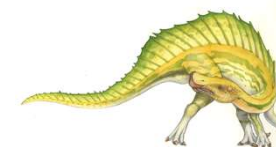


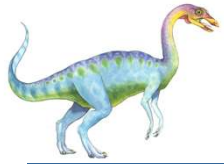


# Escalonamento de Threads

---

- Distinção entre threads em nível de usuário e em nível de kernel
  
- Nos modelos Muitos-para-um e Muitos-para-muitos, a biblioteca de threads escalona threads em nível de usuário para executar em LWP (como kernel threads)
  - Conhecido como **process-contention scope (PCS)** – escopo de contenção do processo - uma vez que a competição do escalamento é dentro do processo
  
- Threads em nível de kernel são escalonadas em CPUs disponíveis. Conhecido como **system-contention scope (SCS)** - escopo de contenção do sistema

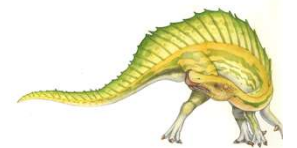


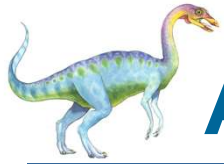


# Escalonamento em Pthread

---

- A API permite especificar entre PCS ou SCS durante a criação da thread
  - PTHREAD SCOPE PROCESS escalona threads usando PCS
  - PTHREAD SCOPE SYSTEM escalona threads usando SCS

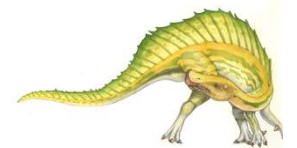


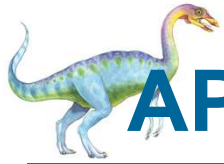


# API de escalonamento na Pthread

---

```
#include <pthread.h>
#include <stdio.h>
#define NUM THREADS 5
int main(int argc, char *argv[])
{
    int i;
    pthread_t tid[NUM THREADS];
    pthread_attr_t attr;
    /* get the default attributes */
    pthread_attr_t attr;
    /* set the scheduling algorithm to PROCESS or SYSTEM */
    pthread_attr_t attr;
    /* set the scheduling policy - FIFO, RT, or OTHER */
    pthread_attr_t attr;
    /* create the threads */
    for (i = 0; i < NUM THREADS; i++)
        pthread_create(&tid[i], &attr, runner, NULL);
}
```

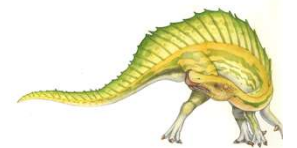




# API de escalonamento na Pthread (Cont.)

---

```
/* now join on each thread */
for (i = 0; i < NUM THREADS; i++)
    pthread join(tid[i], NULL);
}
/* Each thread will begin control in this function */
void *runner(void *param)
{
    printf("I am a thread\n");
    pthread exit(0);
}
```

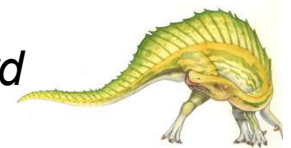




# Escalonamento de Vários Processadores

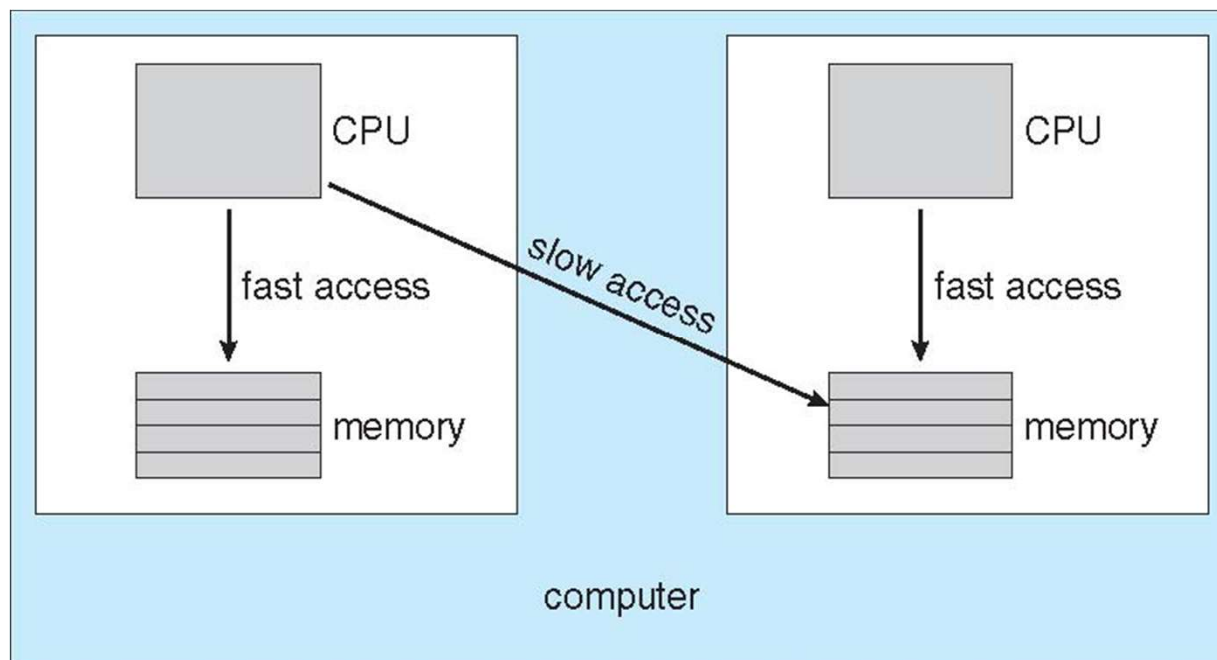
---

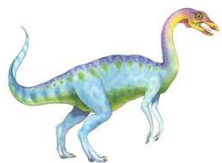
- Escalonamento de CPU é mais complexo quando muitos processadores estão disponíveis
- **Processadores homogêneos** em um multiprocessador
- **Multiprocessamento assimétrico** – somente um processador acessa as estruturas de dados do sistema, aliviando a necessidade de compartilhamento de dados
- **Multiprocessamento simétrico (SMP)** – cada processador é auto-escaloanado, todos os processos em uma fila de processos prontos comun ou cada um tem sua fila privada de processos prontos
- **Afinidade de processador** – processos tem afinidade com processador que está atualmente executando (*soft affinity* x *hard affinity*)





# NUMA e Escalonamento de CPU

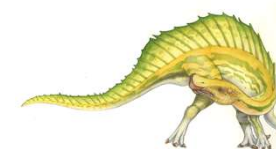


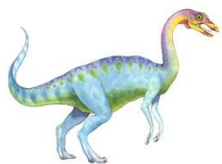


# Processadores Multicore

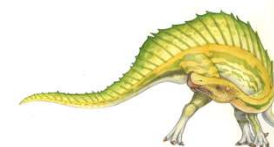
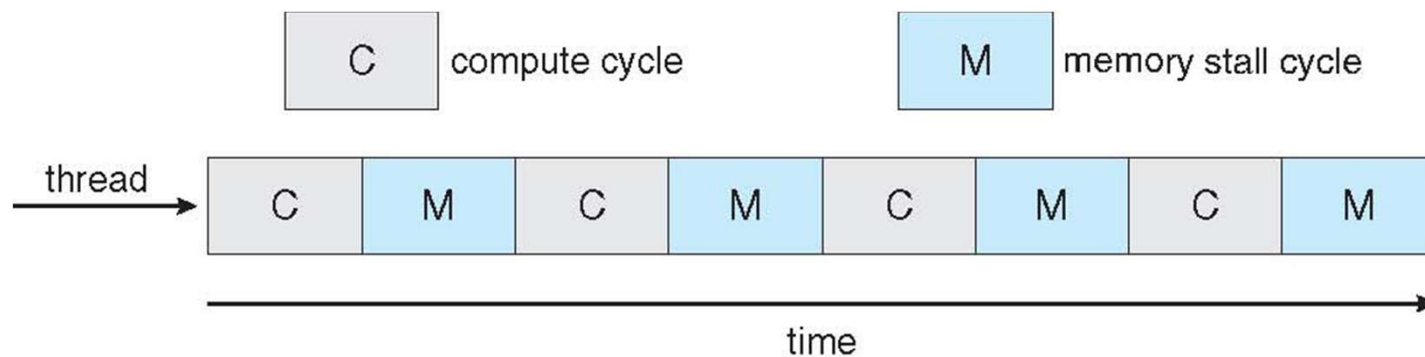
---

- Tendência recente de colocar múltiplos processadores em um único chip
- Mais rápido e consome menos energia
- Múltiplas threads por *core* também em crescimento
  - Considera o atraso no acesso a memória para progredir em outra thread enquanto os dados são transferidos





# Sistema Multithreaded Multicore

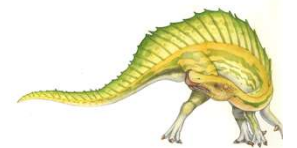


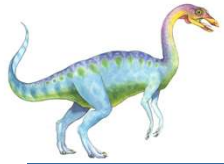


# Exemplos de Sistemas Operacionais

---

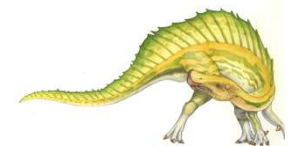
- Escalonamento no Solaris
- Escalonamento no Windows XP
- Escalonamento no Linux





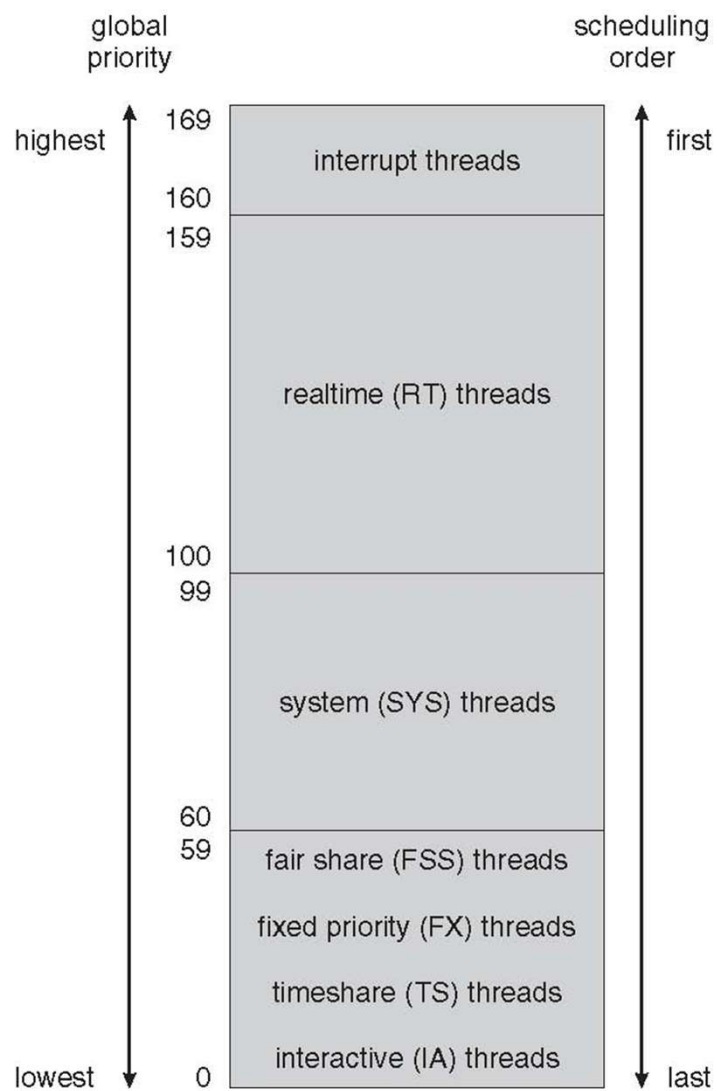
# Tabela de Despacho no Solaris

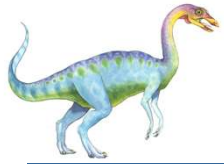
priority	time quantum	time quantum expired	return from sleep
0	200	0	50
5	200	0	50
10	160	0	51
15	160	5	51
20	120	10	52
25	120	15	52
30	80	20	53
35	80	25	54
40	40	30	55
45	40	35	56
50	40	40	58
55	40	45	58
59	20	49	59





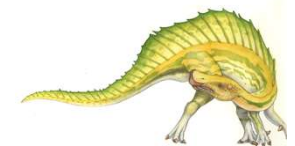
# Escalonamento no Solaris

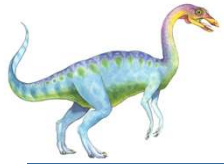




# Prioridades no Windows XP

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

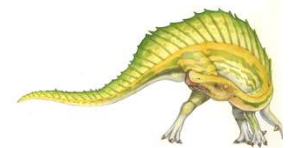




# Escalonamento no Linux

---

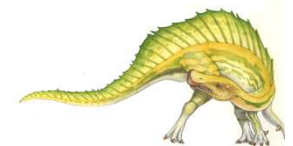
- Tempo de escalonamento de ordem constante  $O(1)$
- Duas escalas de prioridades: tempo compartilhado e tempo real
- Tempo real varia de 0 a 99 e o valor do *nice* de 100 a 140





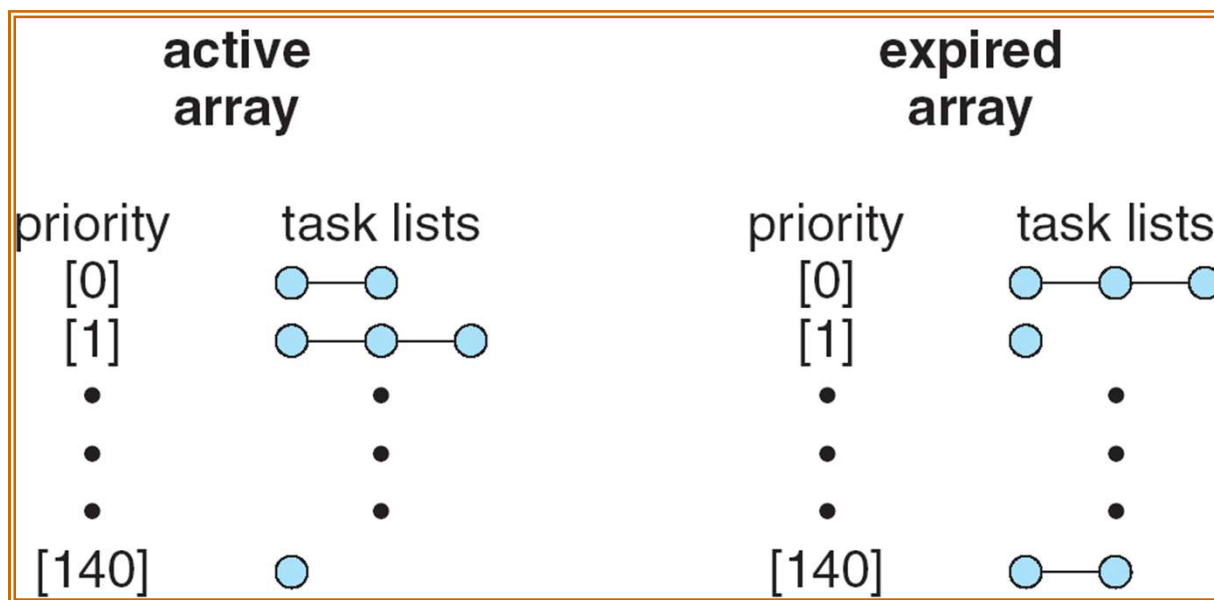
# Relação entre duração de fatia de tempo e prioridades

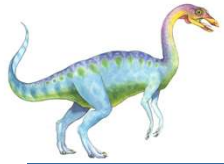
<u>numeric priority</u>	<u>relative priority</u>		<u>time quantum</u>
0	highest	real-time tasks	200 ms
•			
•			
•			
99			
100		other tasks	10 ms
•			
•			
•			
140	lowest		





# Lista de Tarefas Indexada de acordo com Prioridades

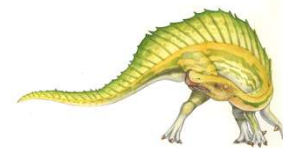




# Avaliação de Algoritmos

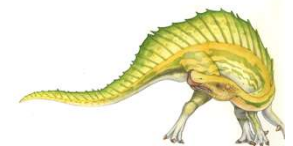
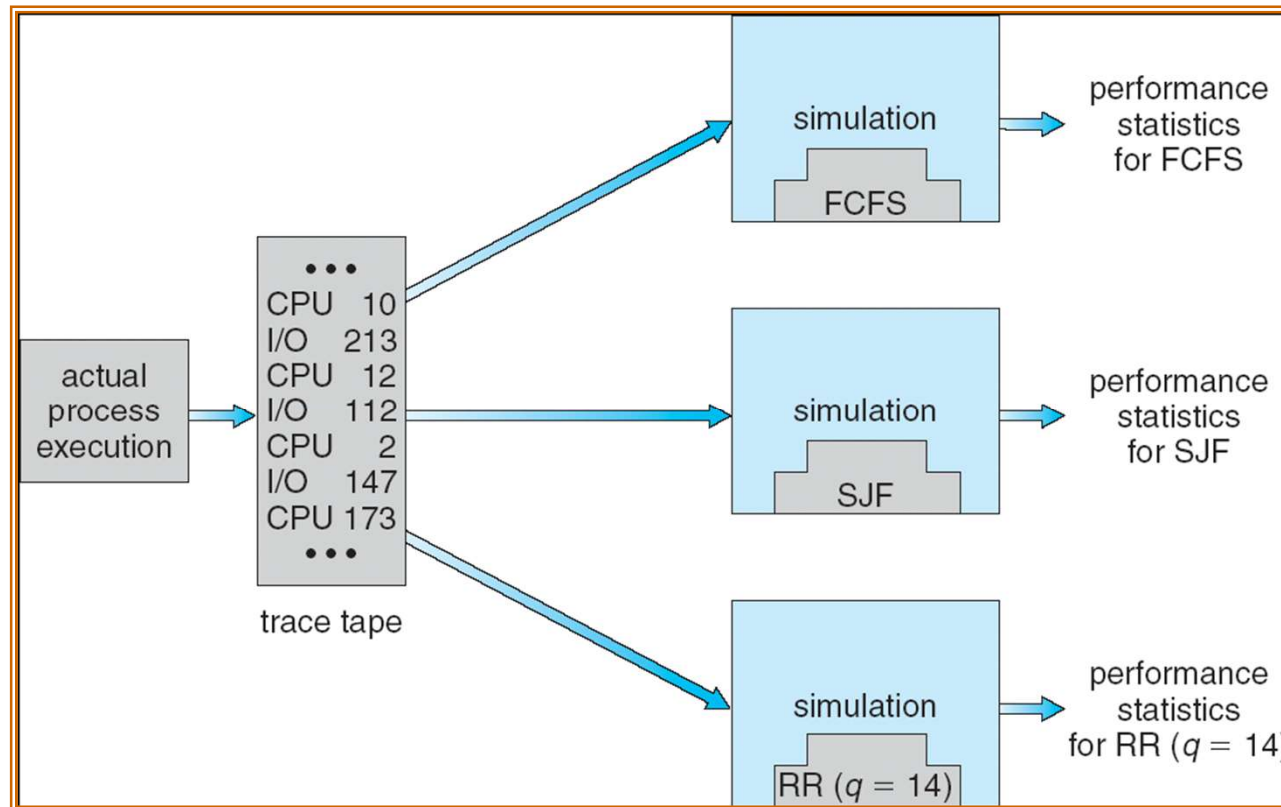
---

- Modelagem determinística – obtém uma carga pré-determinada e define o desempenho de cada algoritmo para esta carga.
- Modelos de Filas
- Implementação





# Avaliação de Escalonadores de CPU por Simulação



# Fim do Chapter 5

---

