Escalonamento da CPU

Escalonamento de CPU

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

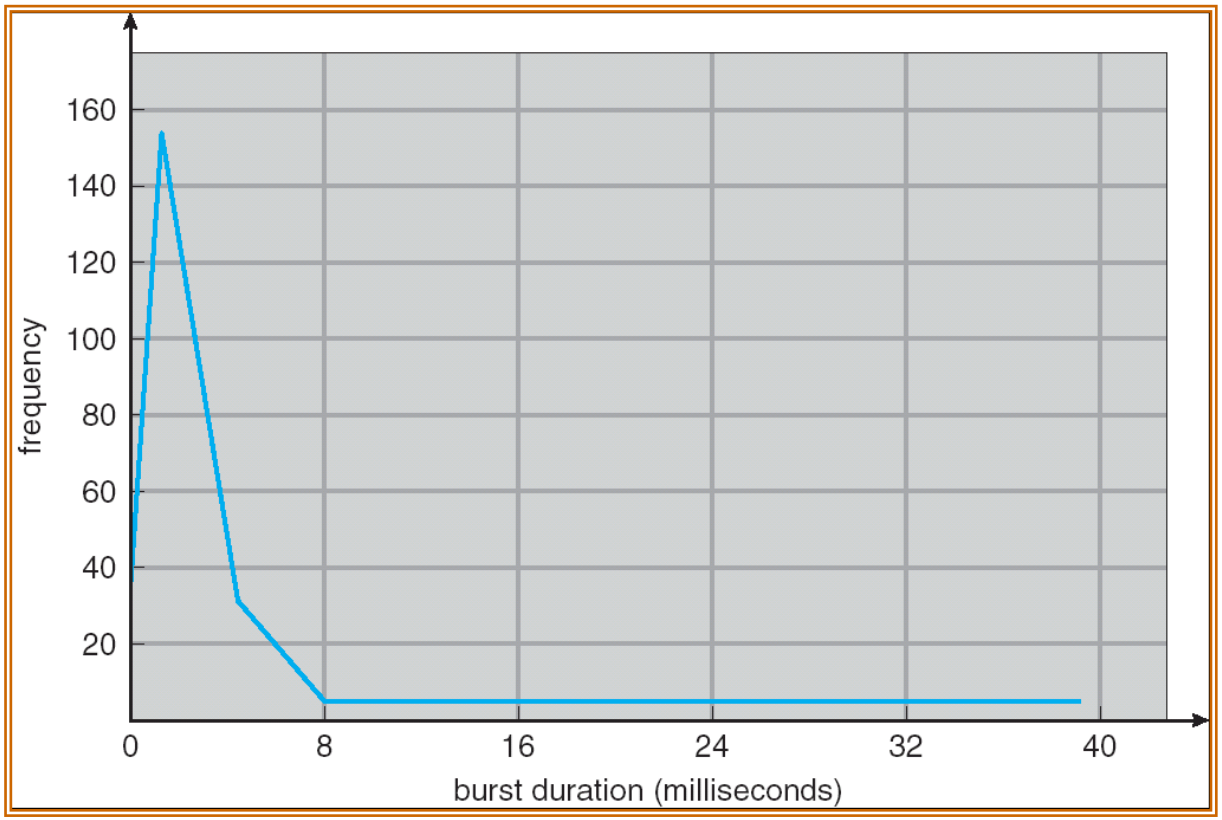
Objetivos

- 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

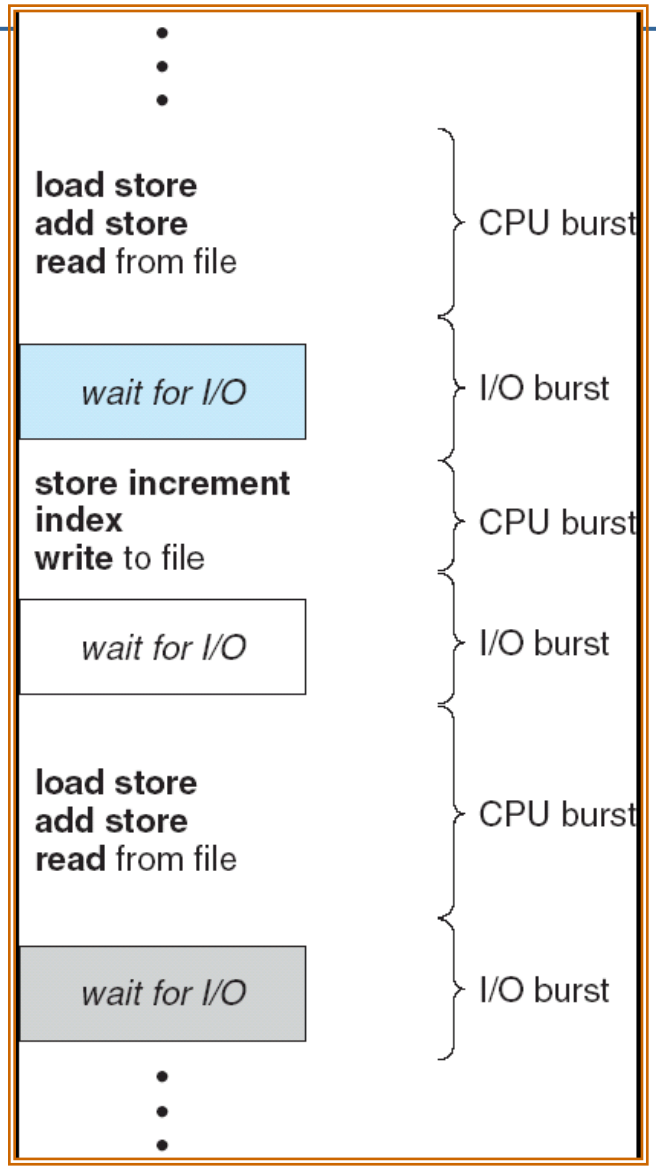
Conceitos Básicos

- 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 ou rajadas de CPU*)

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 *bloqueado*.
  2. Muda do estado *executando* para *pronto*.
  3. Muda do estado *bloqueado* para *pronto*.
  4. Termina.
- Escalonamento nas condições 1 e 4 *é não-preemptivo*.
- Todas as outras alocações são *preemptivas,* ou seja, dependem de uma interrupção de relógio para fazer a troca de contexto e consequente escalonamento

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*)

Processo	Tempo de Rajada
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

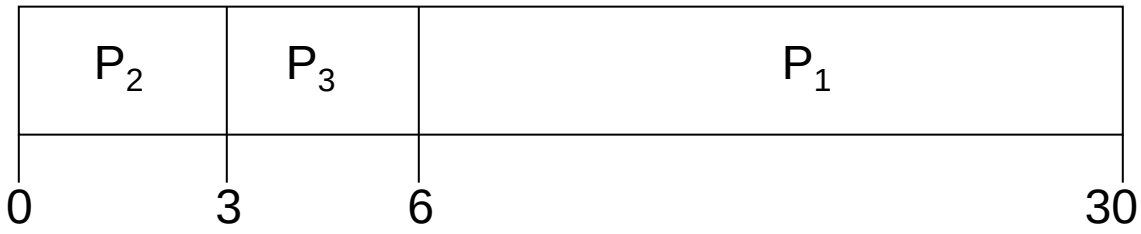
*FCFS - *First Come, First Served*

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 casso anterior.
- *Efeito Comboio:* processos curtos após processos longos

Escalonamento Menor Job Primeiro (SJF*)

- 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.

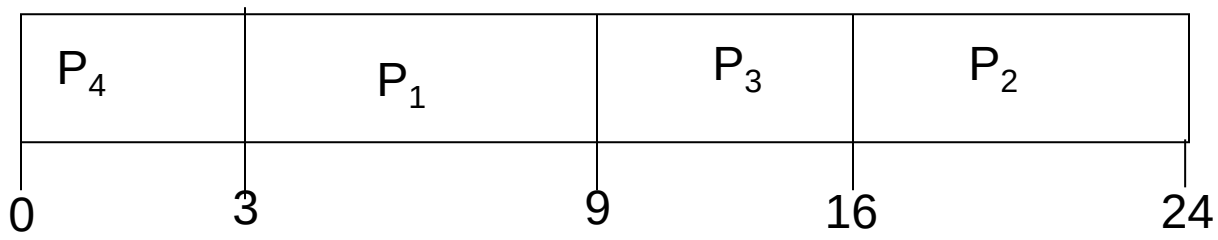
- SJF é ótimo– permite o menor tempo médio de espera para um dado conjunto de processos.
  - A dificuldade é saber o tamanho da próxima requisição de CPU

*SJR - *Shortest-Job-First*

Exemplo de SJF

Processos	Tempo de Rajada
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

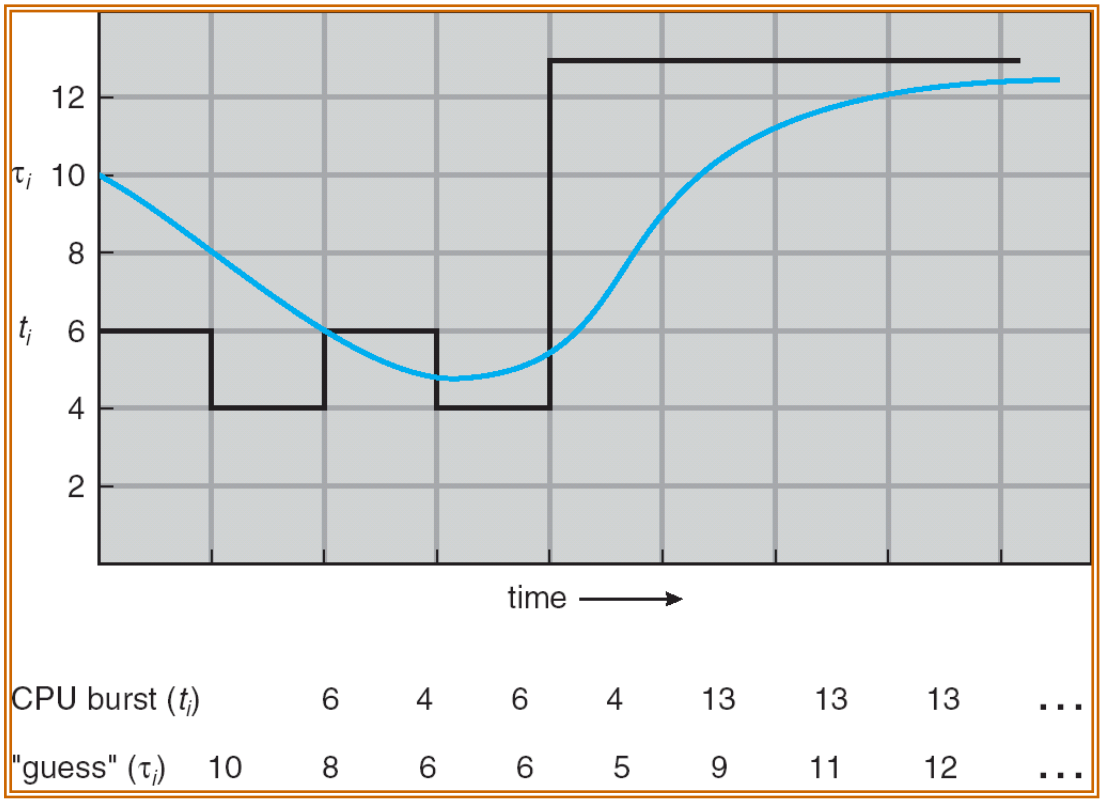
- Pode somente determinar a duração.

- 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. τ_{n+1} = valor previsto para a próxima fase de uso da CPU
  3. $\alpha, 0 \leq \alpha \leq 1$
  4. Define:

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

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:

$$\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\, t_n - 1 + \ldots$$
$$+ (1 - \alpha)^j\, t_n - 1 + \ldots$$
$$+ (1 - \alpha)^{n=1}\, t_n\, \tau_0$$

- Uma vez que tanto α 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 ≡ 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 ≡ *Starvation* (abandono de processo) – processos de baixa prioridade podem nunca executar.
- Solução ≡ *Aging (*envelhecimento*)* – ao passar do tempo, aumentar a prioridade dos processos que ficam em espera no sistema.

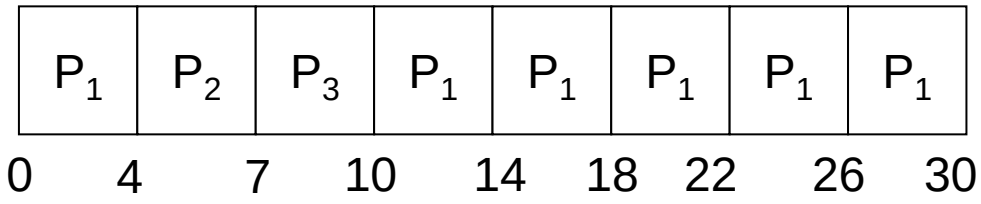
Alocação Circular - Round Robin (RR)

- 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 ou seja FCFS
  - *q* pequeno $\Rightarrow$ *q* deve ser maior que a troca de contexto, senão a sobrecarga é muito alta.

Exemplo: RR com quantum= 4

Processo	Tempo de Rajada
P_1	24
P_2	3
P_3	3

- O diagrama de Gantt é:

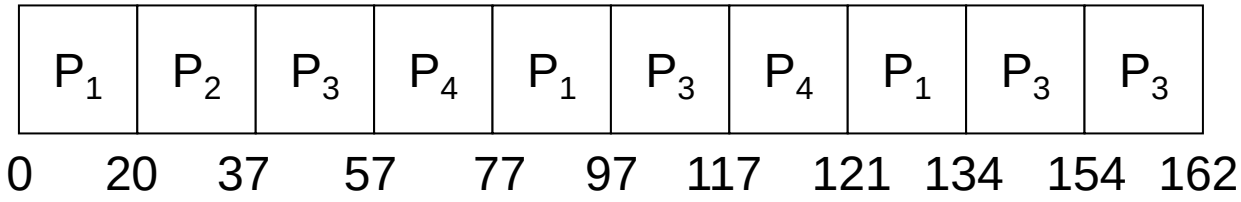


- Tipicamente, possui maior tempo médio de processamento do que o SJF, mas melhor *resposta*.

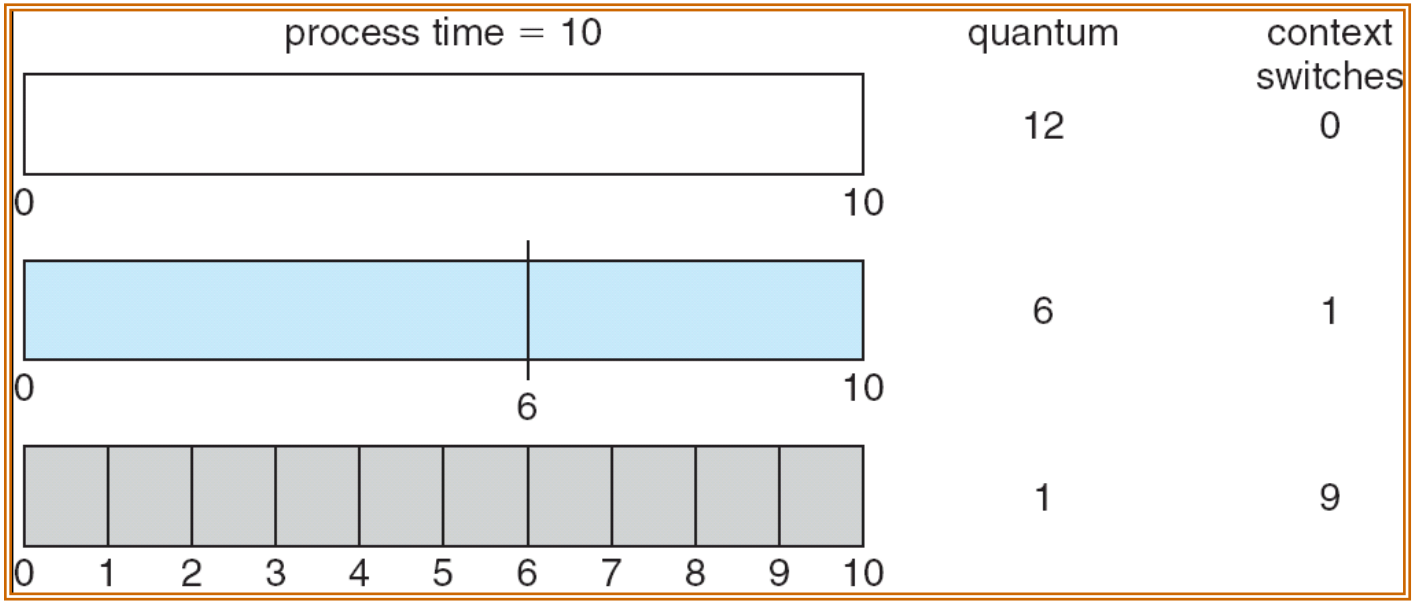
Exemplo: RR com quantum= 20

Processo	Tempo de Rajada
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



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 (lightweight processes ou processos leves)
  - Conhecido como **process-contention scope (PCS)** – escopo de contenção do processo - uma vez que a competição do escalonamento 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 init(&attr);
   /* set the scheduling algorithm to PROCESS or SYSTEM */
   pthread attr setscope(&attr, PTHREAD SCOPE SYSTEM);
   /* set the scheduling policy - FIFO, RR, or OTHER */
   pthread attr setschedpolicy(&attr, SCHED OTHER);
   /* 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-escalonado, todos os processos em uma fila de processos prontos comum 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*)