# Sistemas Operacionais

## Escalonamento

*Parte 1*

Prof. Otávio Gomes

otavio.gomes@unifei.edu.br

# Recapitulando

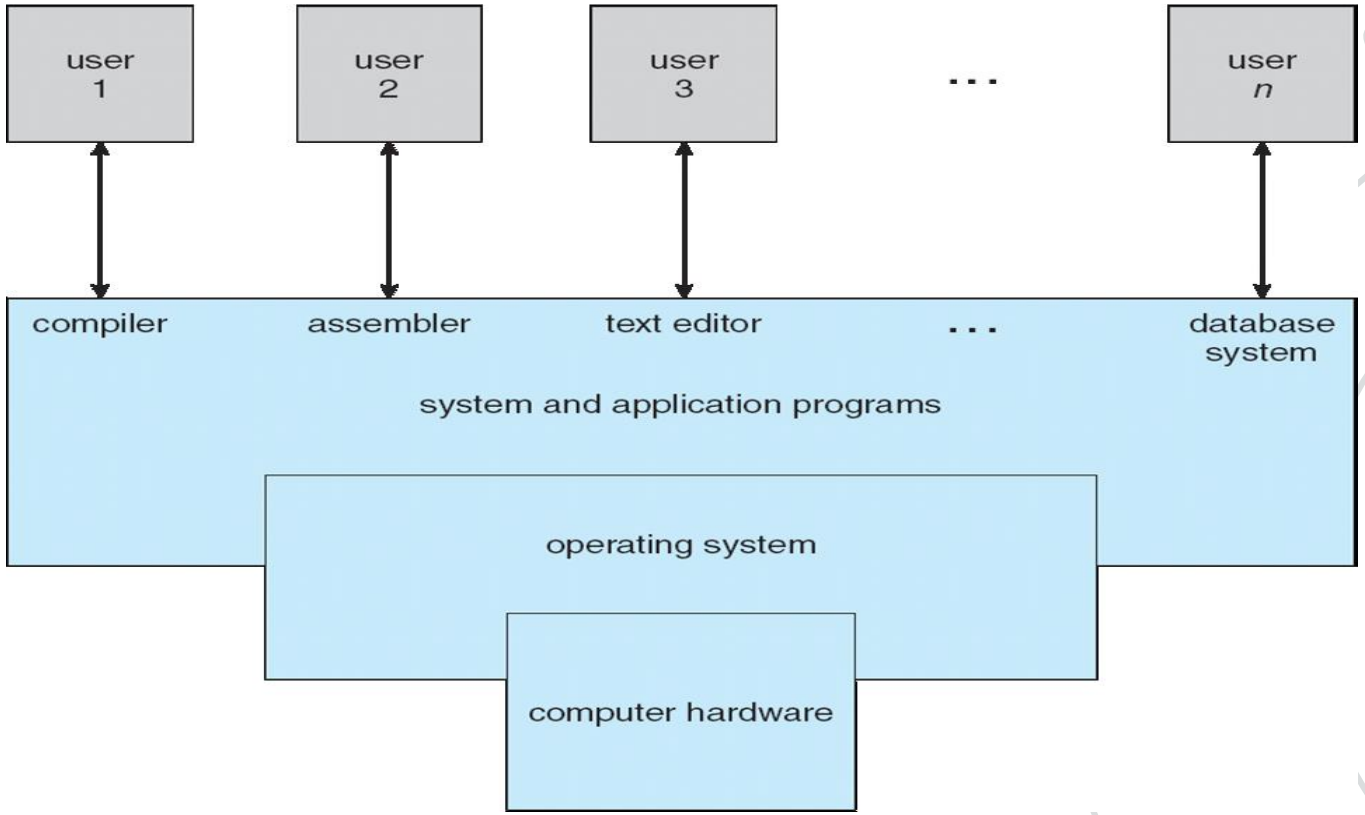
## Sistema Operacional

- Oferece **serviços** de tratamento de interrupções e exceções; gerenciamento de arquivos, memória e dispositivos; suporte à rede; proteção e segurança.
- Realiza **abstração** e **gerência** de recursos limitados.
- Com relação ao tipo de tarefas, podemos classificá-los em:
  - Monotarefa
  - **Multitarefa** (Multiprogramação):
    - Em lote (*Batch*)
    - Tempo Compartilhado (*Time-sharing*)
    - Tempo-Real

# Recapitulando

## Sistema Operacional Multitarefa

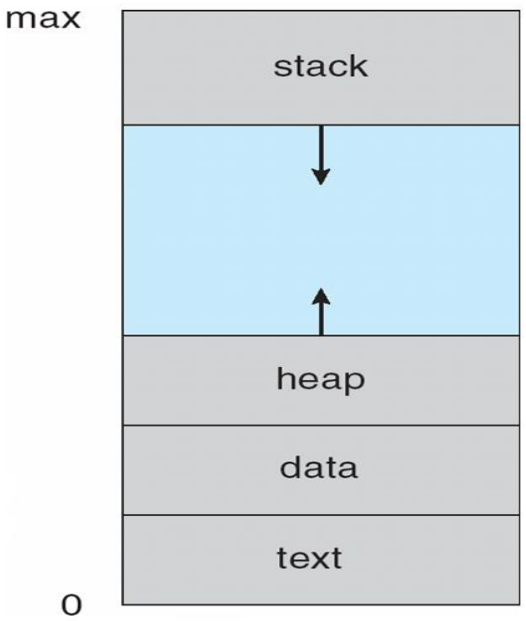
- Possui diversos processos que **competem pela CPU**. Surge, assim, a necessidade de alguma entidade para escalonar a CPU entre os processos.

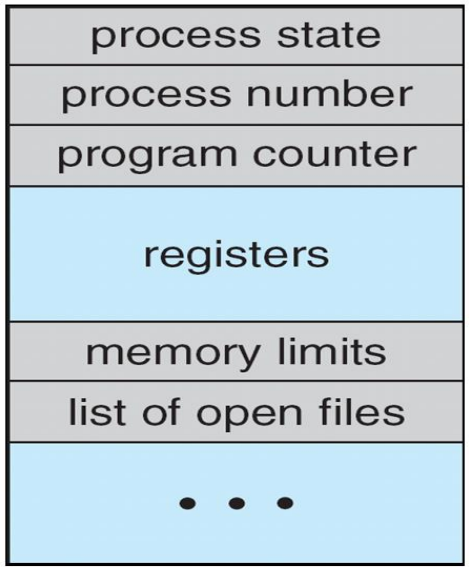


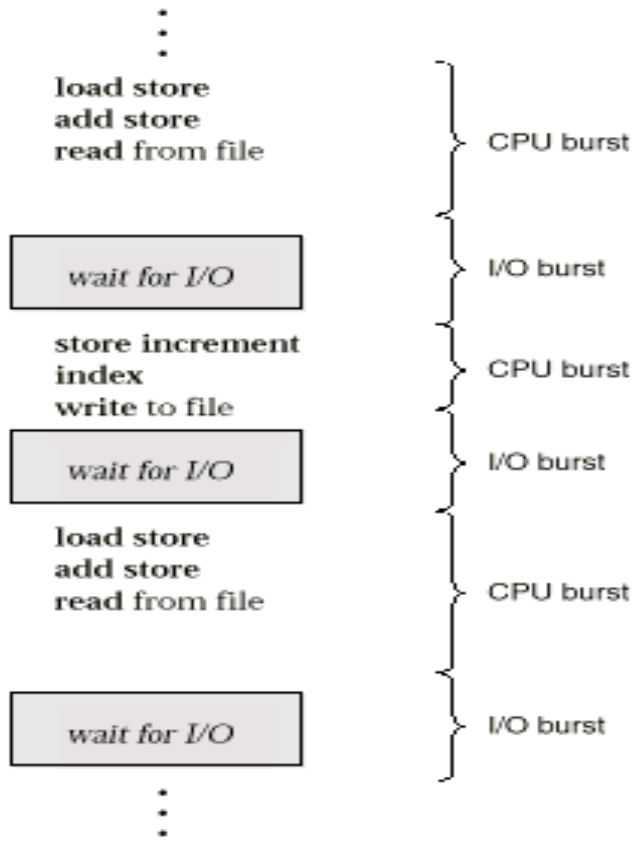
# Recapitulando
## Processos

- É uma instância de um programa em execução;
- Pode ser executado em primeiro ou em segundo plano (*daemon*);
- Pode ser orientado à CPU ou à operações de E/S;
- Possui espaço de endereçamento de memória reservado;
- Possui um Bloco de Controle de Processo (PCB)

que mantém informações relacionadas a seu contexto.



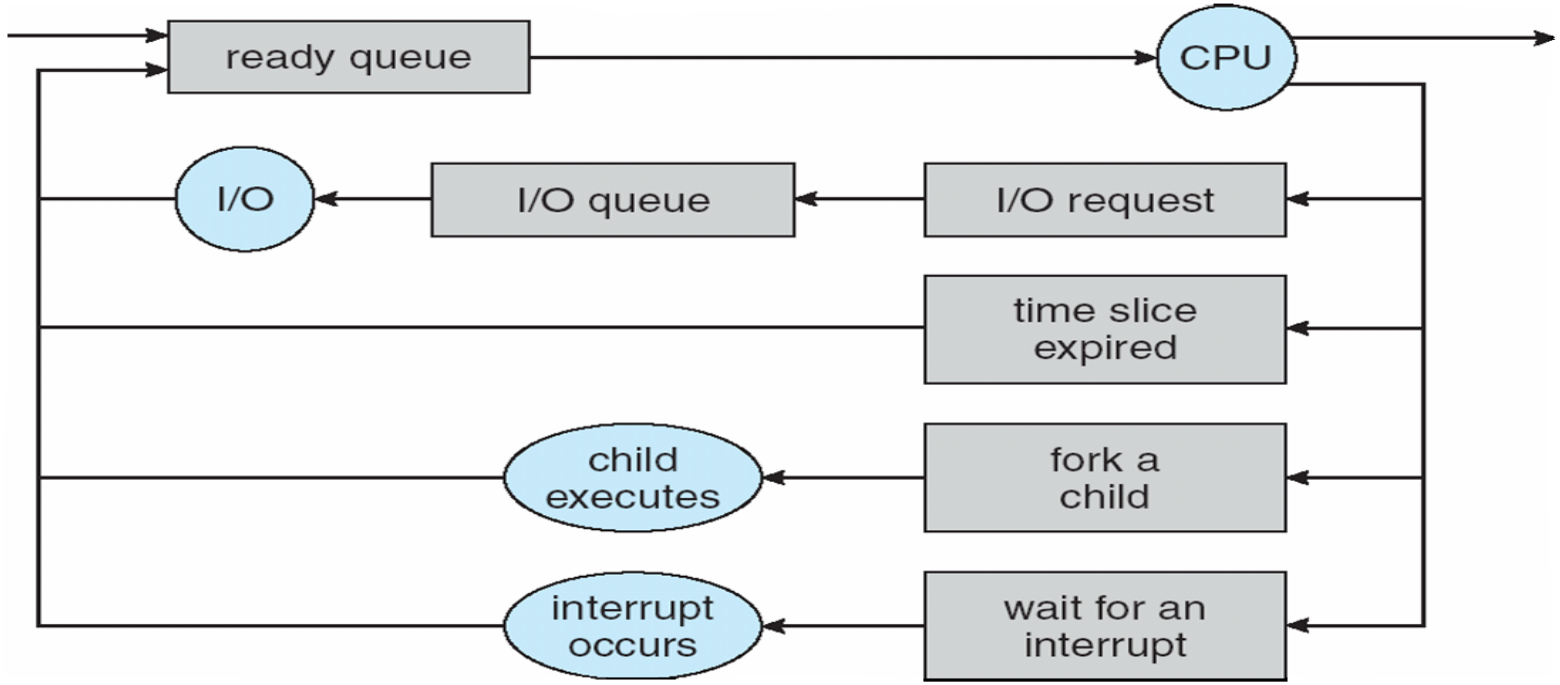



# Recapitulando

## Sistema Operacional Multitarefa

- Possui diversos processos que competem pelo uso da CPU.

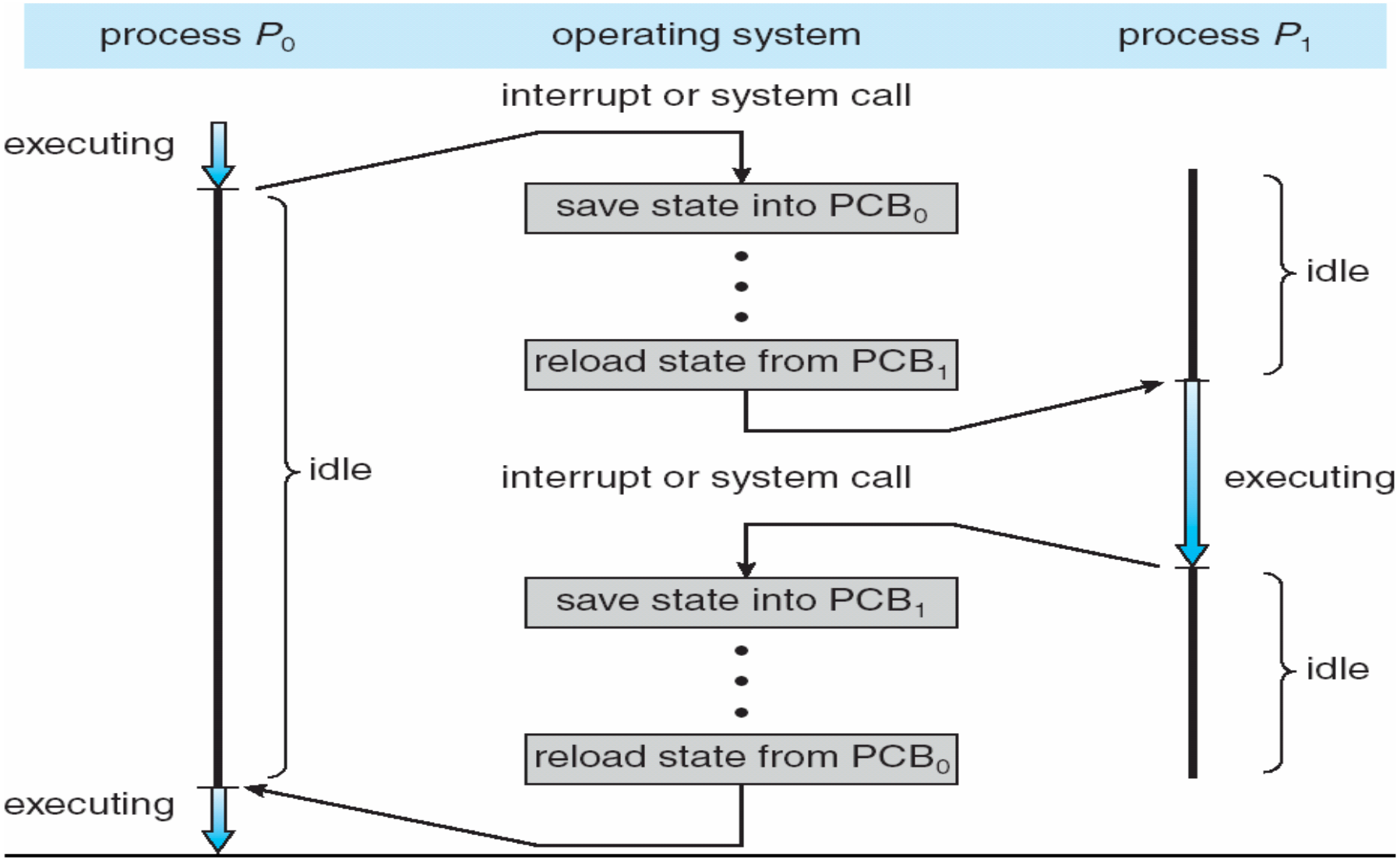



- **Despachante** (*Dispatcher*)**:** módulo que realiza o armazenamento e recuperação dos contextos dos processos e atualiza os PCBs.
- **Escalonador** *(Scheduler):* módulo que controla a mudança de estado dos processos, definindo o próximo processo a ser executado.

# Troca de contexto

- Também conhecido como **chaveamento** ou **mudança** de contexto é o processo computacional de armazenar e restaurar o estado (contexto) de uma CPU de forma que múltiplos processos possam compartilhar uma única instância de CPU.
- Este processo garante que quando o contexto anterior armazenado for restaurado, o ponto de execução voltará ao mesmo estado que foi deixado durante o armazenamento.
- A mudança de contexto leva a uma **sobrecarga** de tempo:
  - É preciso salvar as informações do processo que está deixando/entrando a/na CPU em seu PCB (Bloco de Controle de Processos).
  - Salvar o conteúdo dos registradores.

# Troca de contexto



# Despacho e Escalonamento

## Sistema Operacional Multitarefa

- **Despachante** (*Dispatcher*)**:** módulo que realiza o armazenamento e recuperação dos contextos dos processos e atualiza os PCBs.
  - Armazena e recupera o contexto;
  - Atualiza as informações no PCB;
  - Processo relativamente rápido (0,1ms).

- **Escalonador** *(Scheduler):* módulo que controla a mudança de estado dos processos, definindo o próximo processo a ser executado.
  - Escolhe a próxima tarefa a receber o processador;
  - Parte mais demorada.

# Escalonamento

## Sistema Operacional Multitarefa

- Quando o Escalonador é chamado?
  - Quando um novo processo é **criado**:
    - Por quem ele foi criado? Qual a prioridade deste processo?
    - Quando um processo cria outro, qual executar? Pai ou filho?
  - Um processo chegou ao **fim** e um processo pronto deve ser executado:
    - É necessário definir quem será o próximo da fila.
  - Quando um processo é **bloqueado** (dependência de E/S) e outro deve ser executado.

# Escalonamento

## Sistema Operacional Multitarefa

- Quando ocorre uma resposta de E/S, o escalonador deve:
  - Executar o processo que estava esperando este evento?
  - Continuar executando o processo que já estava sendo executado?
  - Executar um terceiro processo que esteja pronto para ser executado?

# Escalonamento de processos

- Deve possuir um algoritmo que se preocupe com 5 regras:
  - **Justiça** – Todos processos devem ter acesso a CPU (tempo de espera)
  - **Eficiência** – buscar a máxima utilização da CPU
  - Minimizar o **Tempo de Resposta**
  - ***Turnaround*** – Minimiza os usuários *batch*. Tempo para conclusão do processo (alocação + fila + execução CPU + execução E/S)
  - ***Throughput*** – Maximizar o número de *jobs* processados

A partir da finalização da execução de um processo ou de sua parcela de tempo (*quantum*), qual será o novo processo a ser executado?

Um novo processo criado? Um processo que criou outro processo (filho)? Um processo que está pronto há mais tempo?

Como esta escolha pode ser feita?

# Escalonamento de processos

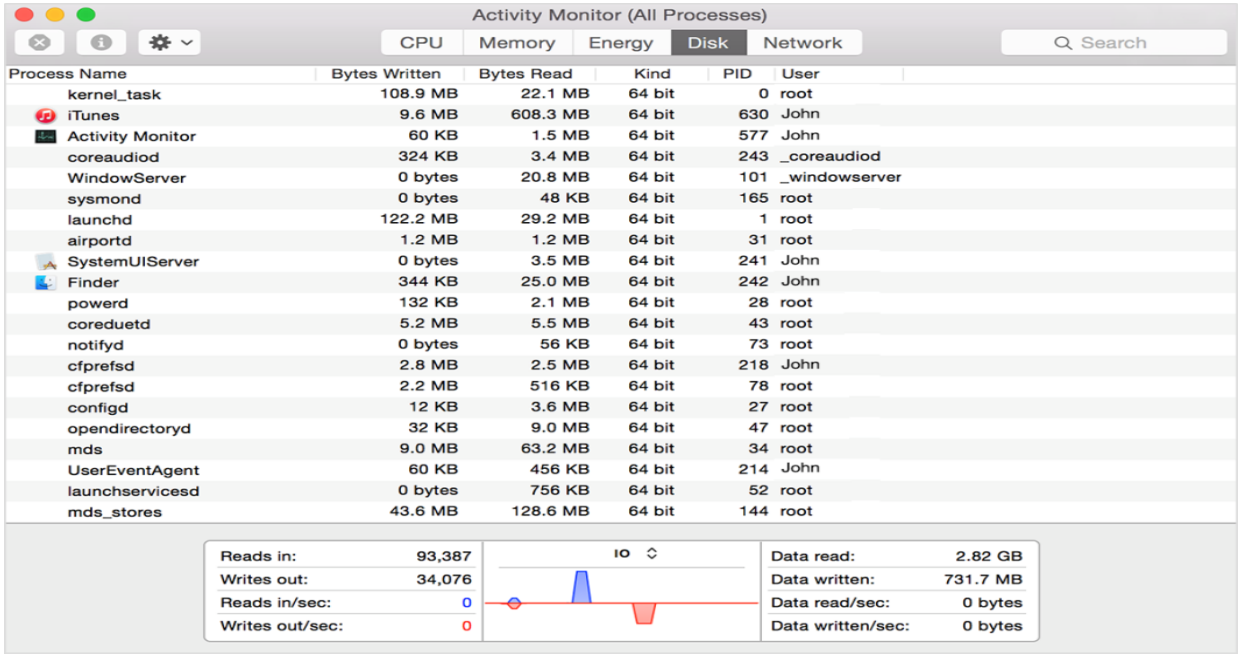
- Com relação ao escalonamento da CPU, pode ser classificado em:
  - **Não-preemptivo:**
    - Implementação mais simples do escalonador.
    - Processo utiliza o processador durante o tempo que quiser.
    - O processo deixa/libera a CPU nas seguintes condições:
      - Término da execução; ou
      - Solicitação de operação de E/S (voluntário).

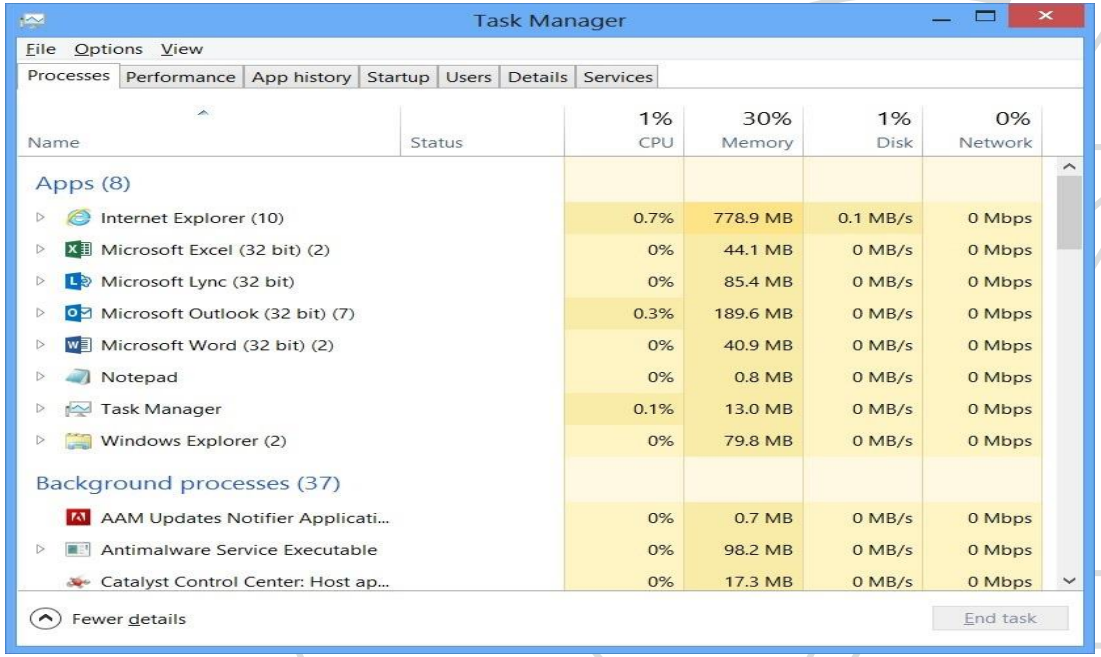
# Escalonamento de processos

- Com relação ao escalonamento da CPU, pode ser classificado em:
  - **Preemptivo:**
    - Escalonador mais complexo.
    - Processo utiliza o processador durante uma parcela de tempo definida (*time-sharing*).
    - Compartilhamento da CPU é garantido. Periodicamente o escalonador interrompe o processo em execução e muda-o para o estado “pronto”.
    - O processo deixa/libera a CPU nas seguintes condições:
      - Término da execução; ou
      - Solicitação de operação de E/S; ou
      - Término do *quantum.*

# Qual o próximo processo a ser enviado à CPU?

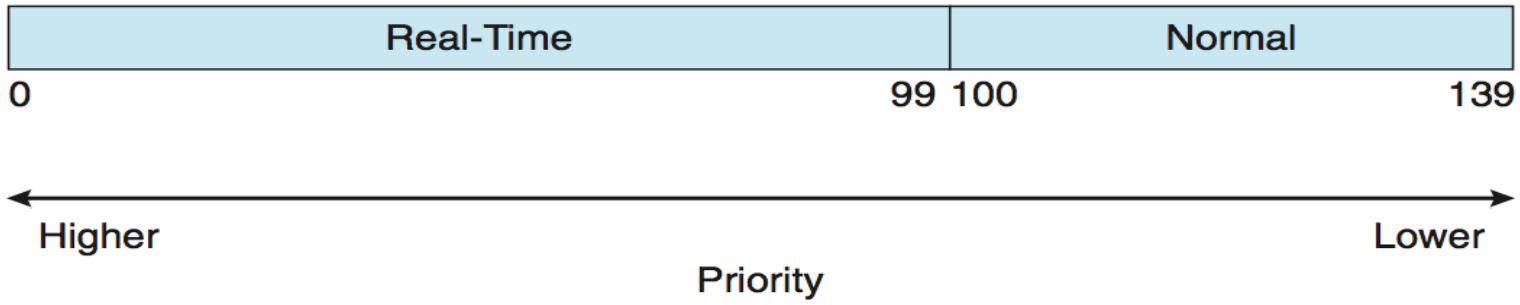
```
top - 19:00:06 up  7:47,  1 user,  load average: 0.65, 0.57, 0.51
Tasks: 198 total,   2 running, 196 sleeping,   0 stopped,   0 zombie
%Cpu(s): 12.6 us,  0.6 sy,  0.0 ni, 86.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  11894.0 total,   1511.5 free,   5763.0 used,   4619.4 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.   5706.3 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 2844 root      20   0 4169800   1.9g 152908 R  46.8  16.4 126:45.88 Web Content
 2758 root      20   0 2560304 462792 162424 S   5.6   3.8  49:01.37 firefox-esr
 1383 root      20   0  521120 113500  82664 S   0.3   0.9  12:35.23 Xorg
 2494 root      20   0 6347740   1.6g  37592 S   0.3  13.7  31:22.93 java
 3030 root      20   0  625936  50372  31888 S   0.3   0.4   0:34.02 gnome-terminal-
11209 root     -51   0   17868   3504   3028 R   0.3   0.0   0:00.03 top
    1 root      20   0  202592   8988   6760 S   0.0   0.1   0:17.10 systemd
    2 root      20   0       0      0      0 S   0.0   0.0   0:00.02 kthreadd
    3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    5 root       0 -20       0      0      0 I   0.0   0.0   0:00.48 kworker/0:0H
    7 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
    8 root      20   0       0      0      0 S   0.0   0.0   0:00.35 ksoftirqd/0
```



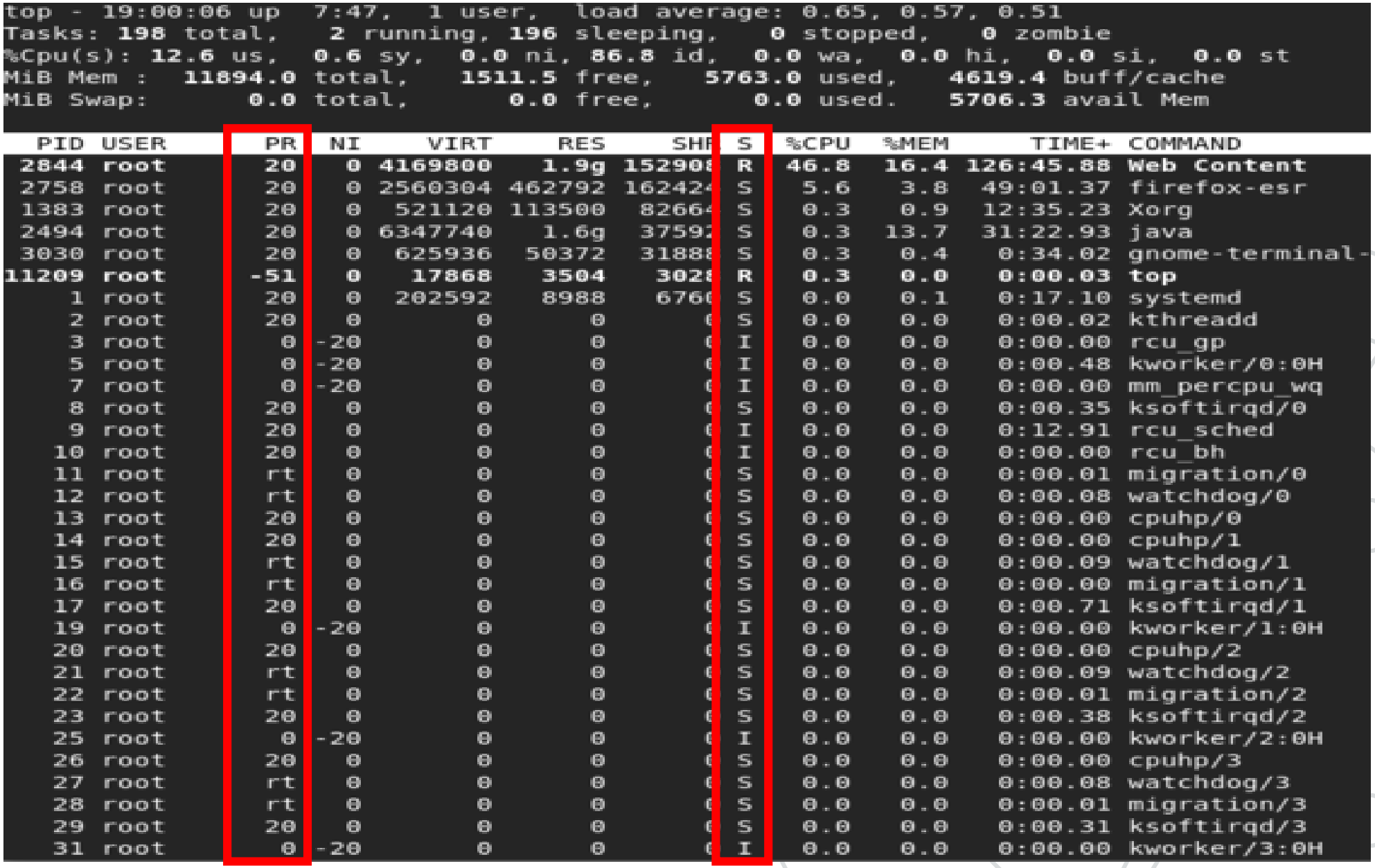



# Processos

- Podem ser descritos como:
  - ***I/O-bound*:** gastam mais tempo fazendo E/S do que computação.
  - ***CPU-bound*:** Gastam mais tempo com computação.
- Podemos classificá-los por:
  - **<u>Uso de recursos:</u>** temos os processos **convencionais** e os de **tempo real** (de sistema).
    - No Linux, os processos de tempo real recebem prioridade entre 1 e 99, enquanto os processos convencionais recebem prioridade entre 100 e 139 (padrão 120).




  - **<u>Tipo de execução:</u>** temos os **interativos**, **em série** ou **tempo real**.




**S** - Em série
**I** - Interativos
**R** - Tempo real

# Algoritmos de escalonamento

- Sistemas Batch
- Sistemas Interativos
- Sistemas Tempo Real

# Algoritmo de escalonamento

Sistemas Batch

# Algoritmo de Escalonamento

## Sistemas *Batch*

- Refere-se a um processamento de dados que ocorre através de um lote de tarefas enfileiradas, de modo que o sistema operacional só processa a próxima tarefa após o término completo da tarefa anterior.

- Estão relacionados ao tempo de *job.*
    1. FCFS (*First-Come, First-Served*)
    2. SJF (*Shortest Job First*)
    3. *Priority-based Scheduling*
    4. SRTN (*Shortest Remaining-Time Next*)

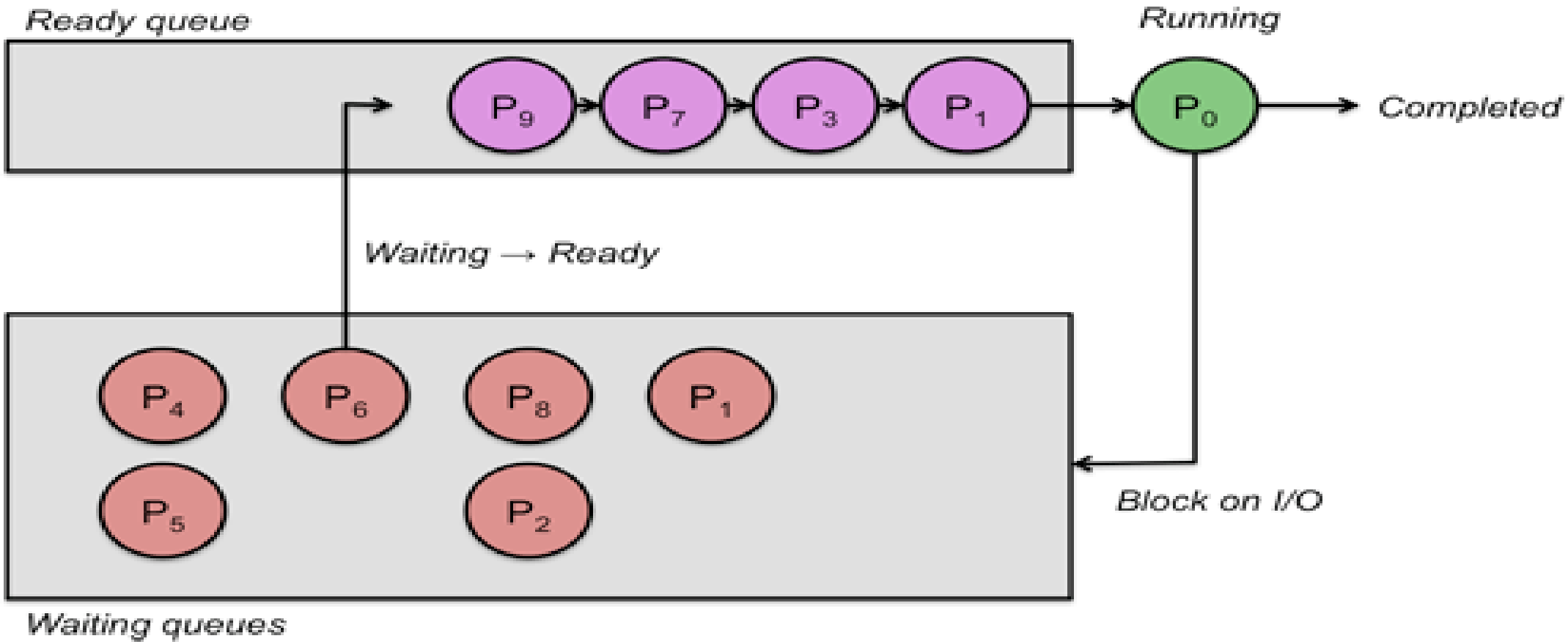
# 1) FCFS (*First-Come, First-Served*)

- **Não-preemptivo;**
- Processos são executados na CPU seguindo a ordem de requisição;
- Fácil de entender e programar (**FIFO**);
- Processos só são interrompidos por uma solicitação de E/S;
- Desvantagem*:*
  - Ineficiente quando há processos que demoram na sua execução.

# Algoritmo de Escalonamento

Sistemas *Batch*

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



# Algoritmo de Escalonamento

Sistemas *Batch*

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

| Process | Arrival Time | Execute Time | Service Time |
|---|---|---|---|
| P0 | 0 | 5 | 0 |
| P1 | 1 | 3 | 5 |
| P2 | 2 | 8 | 8 |
| P3 | 3 | 6 | 16 |

# Algoritmo de Escalonamento

Sistemas *Batch*

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

| Process | Arrival Time | Execute Time | Service Time |
|---|---|---|---|
| P0 | 0 | 5 | 0 |
| P1 | 1 | 3 | 5 |
| P2 | 2 | 8 | 8 |
| P3 | 3 | 6 | 16 |


| Processo | Tempo de espera |
|---|---|
| P0 | 0 - 0 = 0 |
| P1 | 5 - 1 = 4 |
| P2 | 8 - 2 = 6 |
| P3 | 16 - 3 = 13 |

Tempo médio de espera:
(0+4+6+13) / 4 = 5,75

***turnaround time***

# Turnaround Time

- Em sistemas operacionais é o tempo que o S.O. gasta para organizar os processos entre si: requisitar recursos, criar o lote de execução, decidir qual processo vai ser executado.

- Em suma, é o tempo total contado desde a submissão do processo até sua conclusão. Leva em consideração o tempo de espera.

## 2) SJF (***Shortest Job First***) ou SJN (*Shortest Job Next*)

- **Não-preemptivo**;
- Menor processo da lista é executado primeiro;
- Deve-se prever o tempo de execução do processo;
- Menor *turnaround* (médio);
- Desvantagens:
  - Todos os *jobs* precisam ser conhecidos de antemão;
  - Se muitos *jobs* curtos começarem a chegar, os longos podem demorar a ser executados (possibilidade de inanição - *starvation*).

# Algoritmo de Escalonamento

Sistemas *Batch*

## 2) SJF (***Shortest Job First***) ou SJN (*Shortest Job Next*)

| Process | Arrival Time | Execute Time | Service Time |
|---|---|---|---|
| P0 | 0 | 5 | 0 |
| P1 | 1 | 3 | 5 |
| P2 | 2 | 8 | 14 |
| P3 | 3 | 6 | 8 |

P0

# Algoritmo de Escalonamento

Sistemas *Batch*

## 2) SJF (***Shortest Job First***) ou SJN (*Shortest Job Next*)

| Process | Arrival Time | Execute Time | Service Time |
|---|---|---|---|
| P0 | 0 | 5 | 0 |
| P1 | 1 | 3 | 5 |
| P2 | 2 | 8 | 14 |
| P3 | 3 | 6 | 8 |

| P0 |
|---|

| Processo | Tempo de espera |
|---|---|
| P0 | 0 - 0 = 0 |
| P1 | 5 - 1 = 4 |
| P2 | 14 - 2 = 12 |
| P3 | 8 - 3 = 5 |

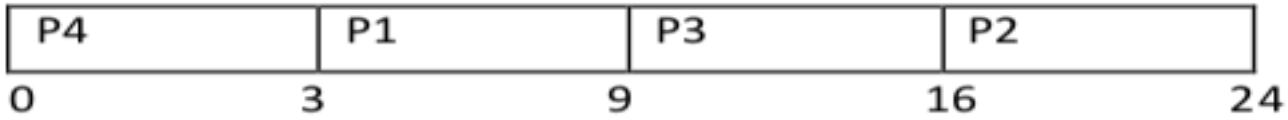
Tempo médio de espera:
(0+4+12+5) / 4 = 5,25

## 2) SJF (***Shortest Job First***) ou SJN (*Shortest Job Next*)

| PID | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time |
|---|---|---|---|---|---|
| P1 | 0 | 6 | 9 | 9 | 3 |
| P2 | 0 | 8 | 24 | 24 | 16 |
| P3 | 0 | 7 | 16 | 16 | 9 |
| P4 | 0 | 3 | 3 | 3 | 0 |

## 2) SJF (***Shortest Job First***) ou SJN (*Shortest Job Next*)

| PID | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time |
|---|---|---|---|---|---|
| P1 | 0 | 6 | 9 | 9 | 3 |
| P2 | 0 | 8 | 24 | 24 | 16 |
| P3 | 0 | 7 | 16 | 16 | 9 |
| P4 | 0 | 3 | 3 | 3 | 0 |



## 3) *Priority-based Scheduling*

- **Não-preemptivo**;
- Cada processo recebe um nível de prioridade;
- Processos com a mesma prioridade são executados de acordo com sua chegada na fila (FCFS / FIFO);
- A prioridade pode ser definida de acordo com requisitos de memória, de tempo ou necessidade de outros recursos.

# 3) *Priority-based Scheduling*

| Process | Arrival Time | Execute Time | Service Time | Priority |
|---|---|---|---|---|
| P0 | 0 | 5 | 0 | 1 |
| P1 | 1 | 3 | 11 | 2 |
| P2 | 2 | 8 | 14 | 1 |
| P3 | 3 | 6 | 5 | 3 |

# 3) *Priority-based Scheduling*

| Process | Arrival Time | Execute Time | Service Time | Priority |
|---|---|---|---|---|
| P0 | 0 | 5 | 0 | 1 |
| P1 | 1 | 3 | 11 | 2 |
| P2 | 2 | 8 | 14 | 1 |
| P3 | 3 | 6 | 5 | 3 |

| Processo | Tempo de espera |
|---|---|
| P0 | 0 - 0 = 0 |
| P1 | 11 - 1 = 10 |
| P2 | 14 - 2 = 12 |
| P3 | 5 - 3 = 2 |

Tempo médio de espera:
(0+10+12+2) / 4 = 6

## 4) SRTN (*Shortest Remaining-Time Next*)

- **Preemptivo** (*quantum*);
- Versão preemptiva do SJF (*Shortest Job First*);
- Processos com menor tempo são executados primeiro.

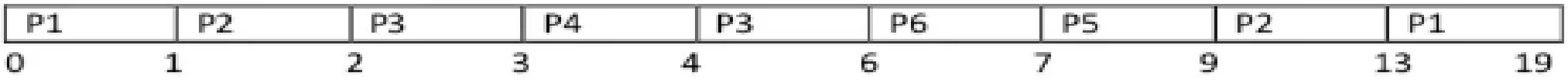
# 4) SRTN (*Shortest Remaining-Time Next*)

| PID | Arrival Time | Burst Time | Completion time | Turn Around time (CT-AT) | Waiting Time (TAT-BT) |
|---|---|---|---|---|---|
| sP1 | 0 | 7 | 19 | 19 | 12 |
| P2 | 1 | 5 | 13 | 12 | 7 |
| P3 | 2 | 3 | 6 | 4 | 1 |
| P4 | 3 | 1 | 4 | 1 | 0 |
| P5 | 4 | 2 | 9 | 5 | 3 |
| P6 | 5 | 1 | 7 | 2 | 1 |


*quantum* = 1

# 4) SRTN (*Shortest Remaining-Time Next*)

| PID | Arrival Time | Burst Time | Completion time | Turn Around time (CT-AT) | Waiting Time (TAT-BT) |
|---|---|---|---|---|---|
| sP1 | 0 | 7 | 19 | 19 | 12 |
| P2 | 1 | 5 | 13 | 12 | 7 |
| P3 | 2 | 3 | 6 | 4 | 1 |
| P4 | 3 | 1 | 4 | 1 | 0 |
| P5 | 4 | 2 | 9 | 5 | 3 |
| P6 | 5 | 1 | 7 | 2 | 1 |



## 4) SRTN (*Shortest Remaining-Time Next*)

- Vantagens:
  - Aumento da vazão com a execução de um maior número de processos em menos tempo;
  - Processos de rajadas (*bursts*) curtas são finalizados rapidamente.

- Desvantagens:
  - Não é possível prever o tempo exato de execução de todos os processos.
  - Processos longos terão sua execução postergada, criando a possibilidade de inanição.

# Bibliografia


- TANENBAUM, Andrew S; BOS, Herbert. Sistemas operacionais modernos. 4a ed. São Paulo: Pearson Education do Brasil, 2016. **Capítulo 2.**

  https://plataforma.bvirtual.com.br/Acervo/Publicacao/1233

- DEITEL, H.M; DEITEL, P.J; CHOFFNES,D.R. Sistemas Operacionais. 3a ed. São Paulo: Pearson Prentice Hall, 2005. **Capítulo 8.**

  https://plataforma.bvirtual.com.br/Acervo/Publicacao/315

# Sistemas Operacionais

Prof. Otávio Gomes

otavio.gomes@unifei.edu.br