

# Sistemas Operacionais

Sistemas de Informação

Faculdade Sul-Americana - FASAM



Airton Bordin Junior

airtonbjunior@gmail.com



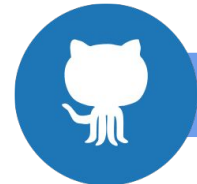
# Quem sou eu?

## Airton Bordin Junior

- Graduação
  - Ciência da Computação
  - Gestão Pública
- Especialização
  - Redes de Computadores
  - MBA Gerenciamento de Projetos
- Mestrado em Ciência da Computação - Inteligência Artificial



[airtonbjunior@gmail.com](mailto:airtonbjunior@gmail.com)



[github.com/airtonbjunior](https://github.com/airtonbjunior)



[linkedin.com/in/airtonbjunior](https://linkedin.com/in/airtonbjunior)



Baixe a apresentação

# Ementa

- Conceitos básicos de sistemas operacionais: processos, organização de sistemas operacionais, chamadas de sistema;
- Gerência do processador: estados do processo, escalonamento;
- Entrada e saída: dispositivos controladores, softwares de E/S, interrupções, dependência e independência;
- Gerência de Memória: partições fixas e variáveis, paginação, segmentação, memória virtual;
- Estudos de Caso: Famílias Windows e GNU/Linux.

# Bibliografia básica



- DEITEL, Harvey M.; DEITEL, Paul J.; CHOFFNES, David R. Sistemas operacionais. 3. ed. São Paulo: Pearson, 2005
- SILBERSCHATZ, Abraham; GALVIN, Peter Baer; GAGNE, Greg. Fundamentos de sistemas operacionais – princípios básicos. 9. ed. Rio de Janeiro: LTC, 2013
- TANENBAUM, Andrew S. Sistemas operacionais modernos. 3. ed. São Paulo: Pearson, 2008

# Bibliografia complementar



- NEMETH, Evi; SNYDER, Gary; HEIN, Trent R. Manual completo de linux - guia do administrador. 2. ed. São Paulo: Pearson, 2007
- OLIVEIRA, Rômulo Silva de.; TOSCANI, Simão Sirineo; CARISSIMI, Alexandre da silva. Sistemas operacionais. 4. ed. São Paulo: Bookman, 2010
- STALLINGS, William. Arquitetura e organização de computadores. 8. ed. São Paulo: Pearson, 2010
- TANENBAUM, Andrew S. e WOODHULL, Albert S. Sistemas operacionais: projeto e implementação. 3. ed. São Paulo: Bookman, 2008
- TANENBAUM, Andrew S.; AUSTIN, Todd. Organização estruturada de computadores. 6. ed. São Paulo: Pearson, 2013

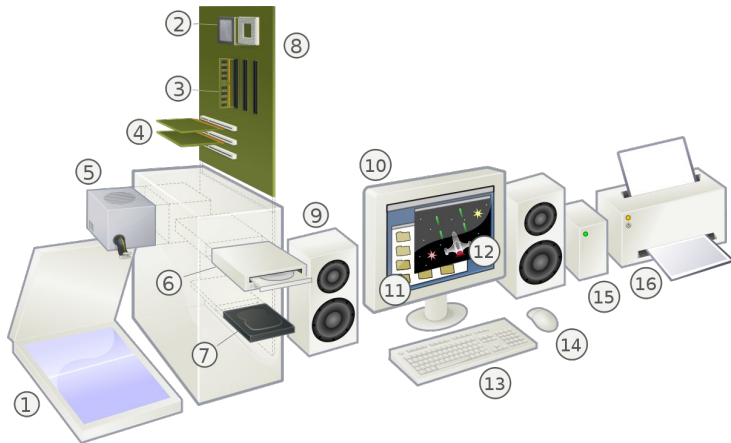
# Avaliação



- Média aritmética de duas notas semestrais: N1 e N2;
- Nota 1
  - Avaliação escrita: 6,0 pontos;
  - Outras atividades: 4,0 pontos.
- Nota 2
  - Avaliação escrita: 6,0 pontos;
  - DIA: 1,0 ponto;
  - Outras atividades: 3,0 pontos.

# Introdução

- Computador é um sistema complexo;
- E se todo programador tivesse de compreender como todas as partes funcionam em detalhe?
- E se fosse preciso escrever código diferente para cada marca/modelo de dispositivo?



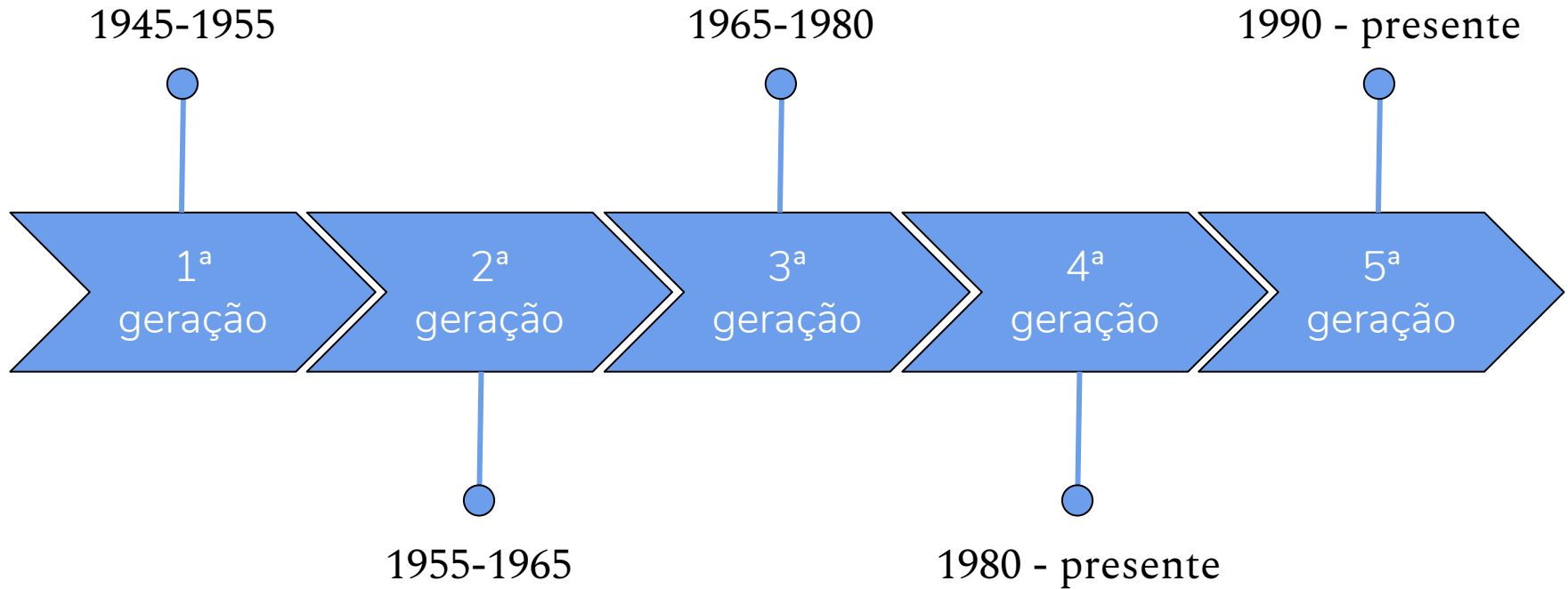
Qual a taxa de atualização do monitor?  
Qual a velocidade máxima de rotação do HD?  
Qual o endereço físico da variável abc?



- Qual a definição de um Sistema Operacional?
- Duas funções essenciais
  - Fornecer um conjunto de recursos abstratos (ao invés de recursos confusos de *hardware*);
  - Gerenciar recursos de *hardware*.

# Histórico

# Histórico





- Válvulas (*Bugs*);
- Cálculos matemáticos;
- Grupo restrito de usuários;
- Necessário conhecimento do funcionamento do *hardware*;
- “Inexistência” de Sistemas Operacionais.

1ª  
geração

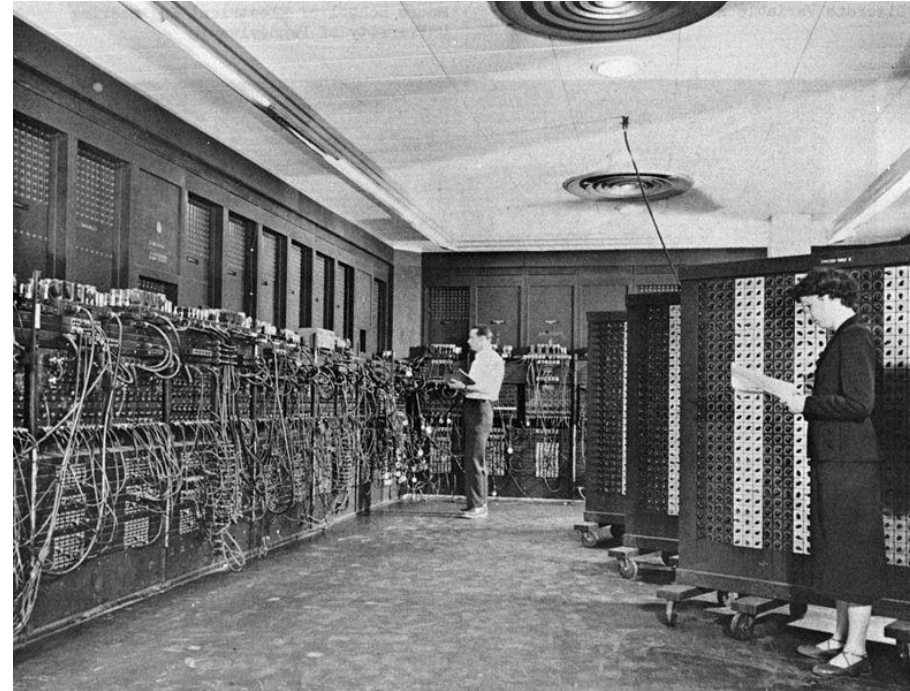
2ª  
geração

3ª  
geração

4ª  
geração

5ª  
geração

- ENIAC
  - ~17.500 válvulas;
  - 30 toneladas;
  - 180 m<sup>2</sup>;
  - 200 bits RAM.



<https://tecnoblog.net/56910/eniac-primeiro-computador-do-mundo-completa-65-anos/>

# Histórico

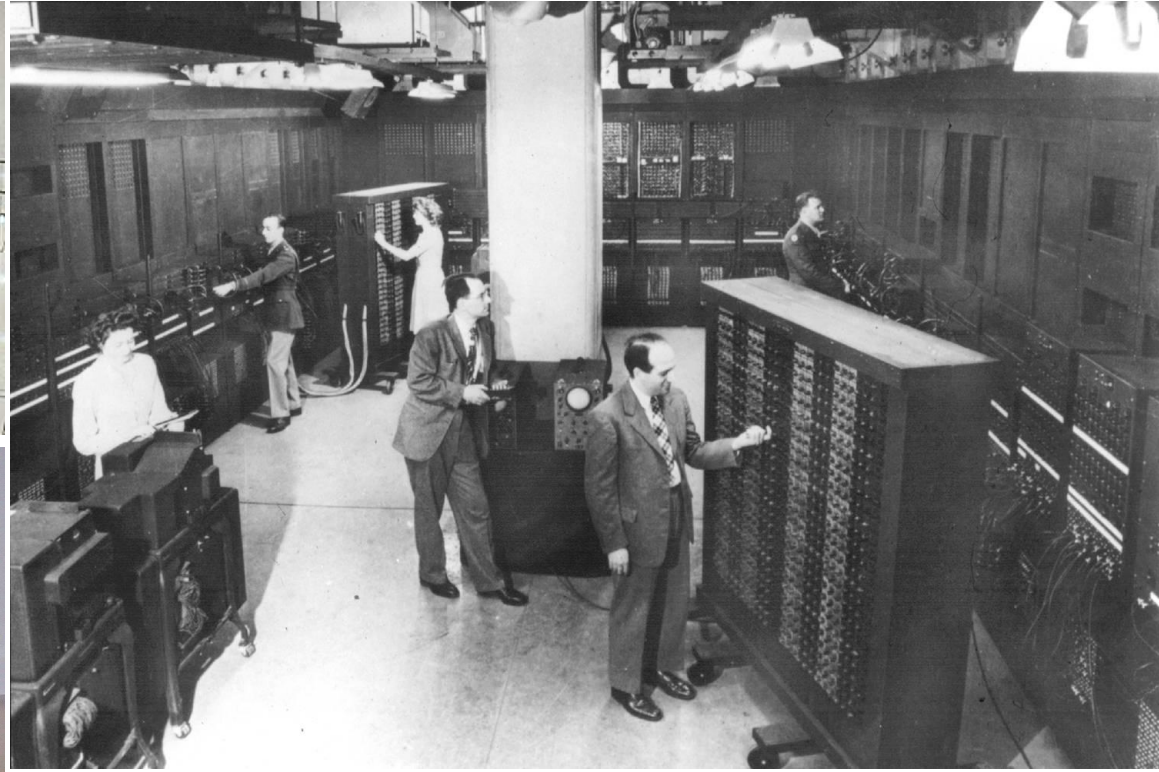
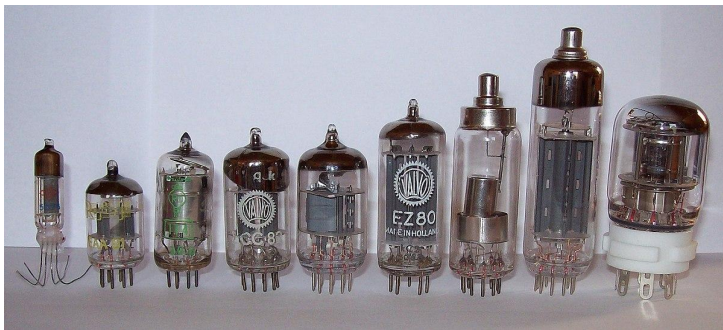
1ª  
geração

2ª  
geração

3ª  
geração

4ª  
geração

5ª  
geração



1ª  
geração

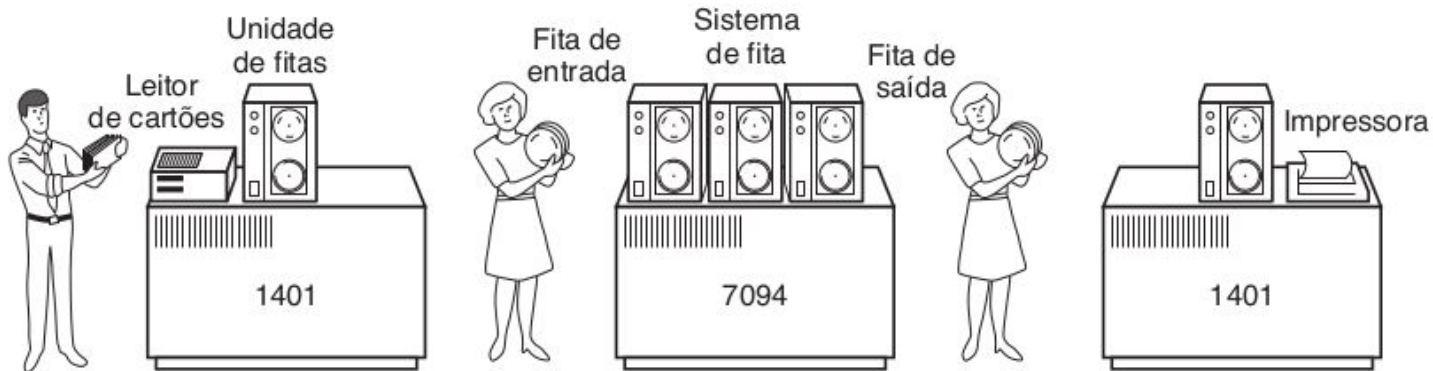
2ª  
geração

3ª  
geração

4ª  
geração

5ª  
geração

- Transistores;
- Computadores mais rápidos e eficientes;
- Cartões perfurados, sistemas em lote;
- *Assembly*.





# Histórico

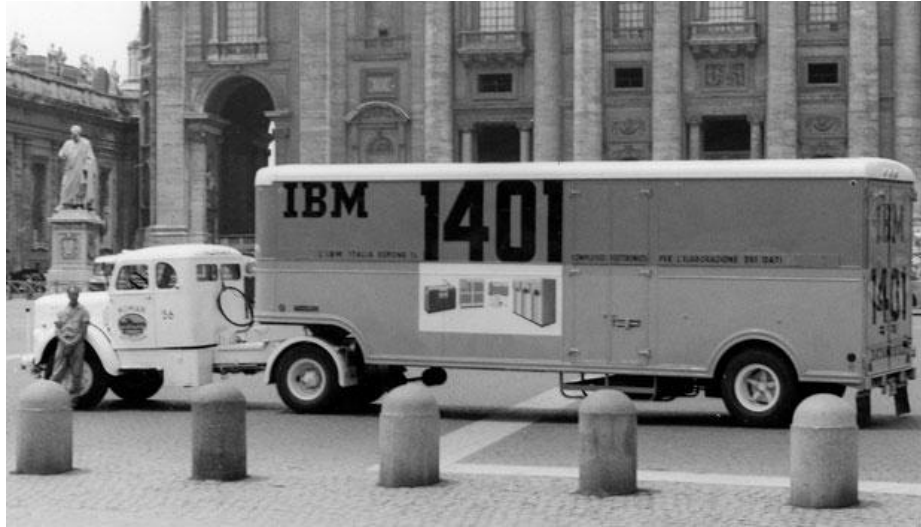
1ª  
geração

2ª  
geração

3ª  
geração

4ª  
geração

5ª  
geração



IBM 1401: The Mainframe  
<https://www.ibm.com/ibm/history/ibm100/us/en/icons/mainframe/impacts/>





- Circuitos integrados;
- Multiprogramação;
- Linha de produtos distintos e incompatíveis;
- IBM 360 - OS/360;
- “Popularização” das linguagens de alto nível.

# Histórico

1ª  
geração

2ª  
geração

3ª  
geração

4ª  
geração

5ª  
geração





- Computadores pessoais, microprocessadores;
- Processadores Intel 8086;
- IBM PC;
- MS-DOS (*Bill Gates*), UNIX, GNU/Linux, OS X;
- Linguagens de alto nível;
- Periféricos avançados e GUI;
- Redes de computadores.

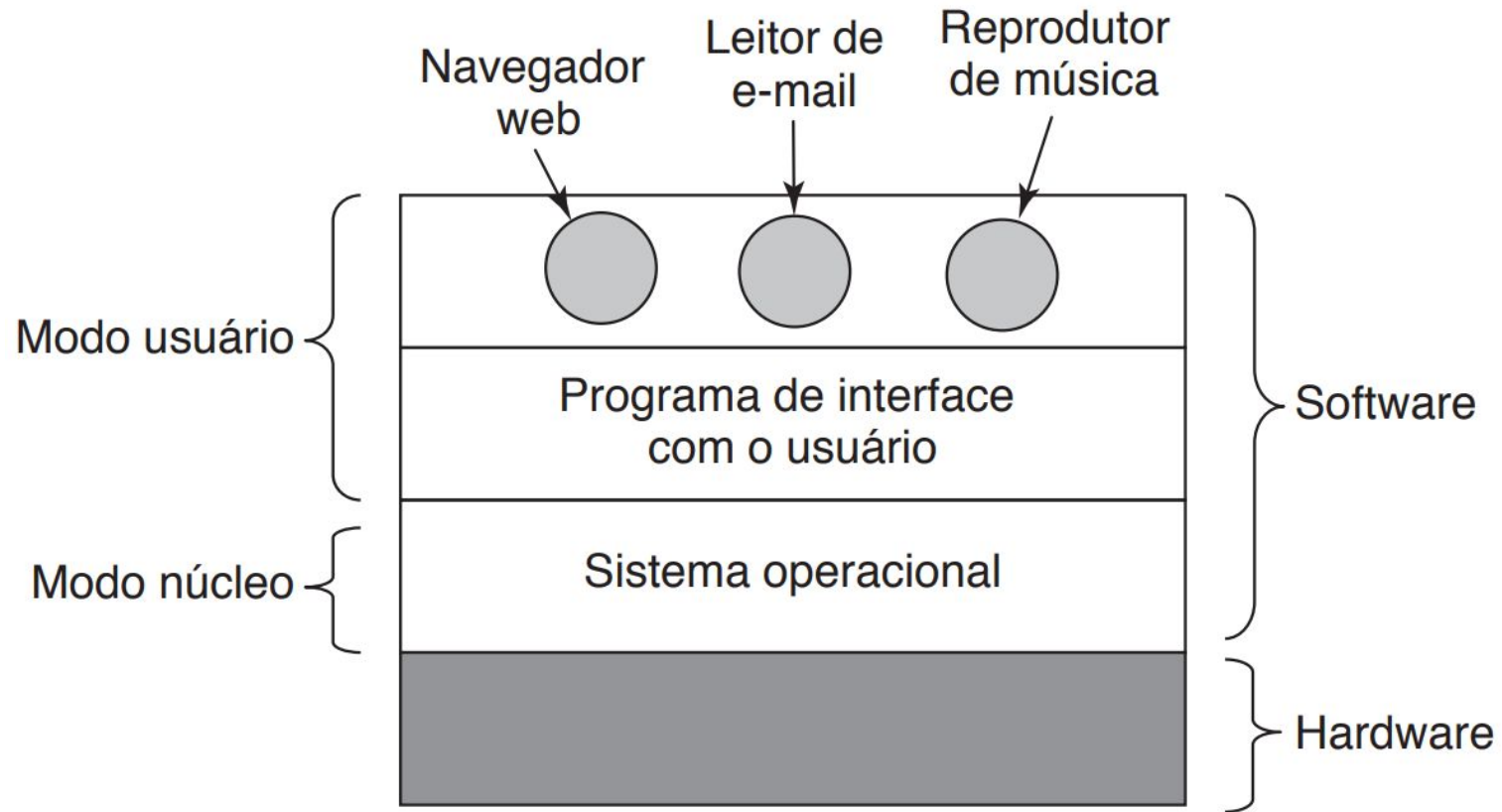


- Computadores móveis;
- Blackberry OS;
- Symbian;
- Android;
- iOS.

# Principais conceitos

- O que é um Sistema Operacional?
  - Um programa de computador (software) que fica entre as aplicações e o equipamento (hardware);
  - É um software que habilita as aplicações e interage com o *hardware* do computador;

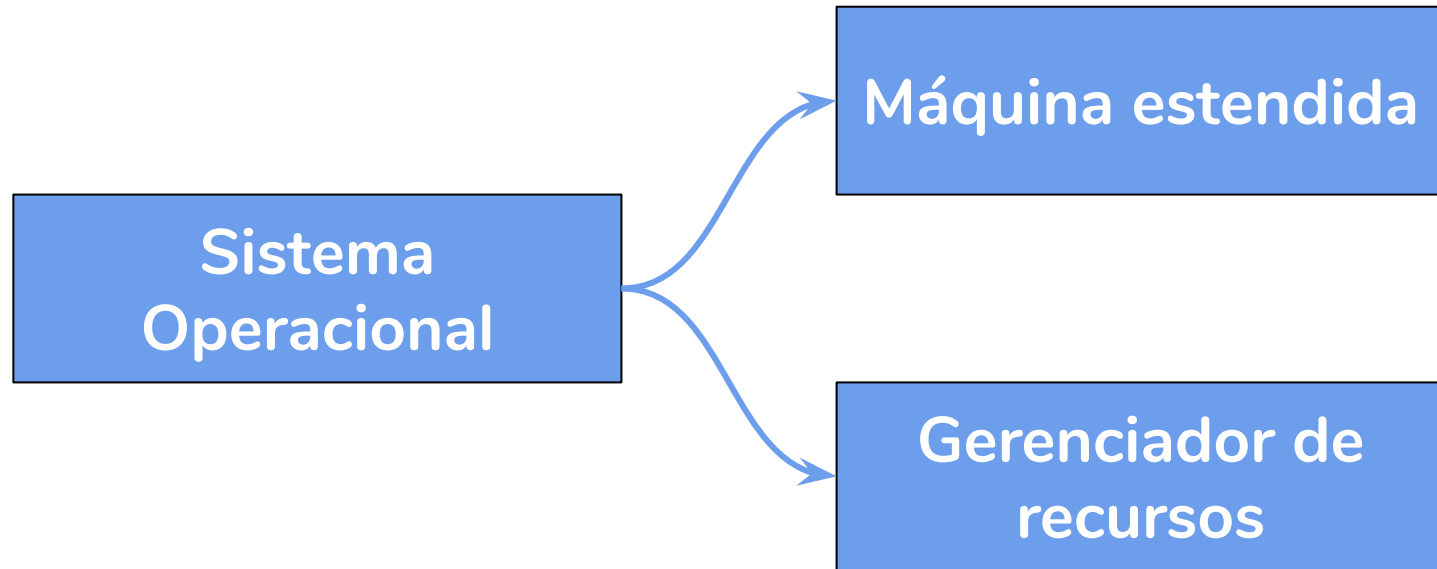
# Principais conceitos



- Qual o objetivo de um Sistema Operacional?
  - Gerenciar os componentes de *hardware* e fornecer aos programas de usuário uma interface com o *hardware* mais simples;
  - Utilizar *hardware* e *software* de maneira eficiente.



- Gerenciamento do processador;
- Gerenciamento de memória;
- Gerenciamento de dispositivos;
- Gerenciamento de arquivos;
- Gerenciamento de proteção;
- Suporte de redes;
- Outros suportes.



- S.O. como máquina estendida
  - O S.O. é a camada de software que oculta a complexidade do *hardware*;
  - Facilita a criação de novos sistemas, de forma que o desenvolvedor não necessite saber detalhes de funcionamento dos periféricos.

- S.O. como gerenciador de recursos
  - O S.O. é responsável por organizar e alocar de forma ordenada e eficiente os recursos de *hardware* do computador;
  - Definir políticas de uso dos recursos de hardware pelas aplicações e resolver disputas e conflitos que venham a ocorrer.

- Tipos de Sistemas Operacionais
  - Monotarefa;
  - Multitarefa
    - Batch;
    - Tempo compartilhado;
    - Tempo real.
  - Multiprocessadores.

# Gerenciamento do Processador

- Primeiros S.O. eram caracterizados por apenas um programa sendo executado por vez;
- Computadores modernos = várias tarefas ao mesmo tempo
  - Processador é alternado entre um programa e outro;
  - Pseudoparalelismo.

- Para que isso seja possível, é necessário um monitoramento complexo de todas as atividades;
- Foi desenvolvido um modelo que tornou o “paralelismo” mais fácil de manipular: modelo de **PROCESSOS**.





O conceito mais central em qualquer sistema operacional é o processo: uma abstração de um programa em execução. Tudo o mais depende desse conceito, e o projetista (e estudante) do sistema operacional deve ter uma compreensão profunda do que é um processo o mais cedo possível [...] Sem a abstração de processo, a computação moderna não poderia existir



Tanenbaum, Andrew S



O conceito mais central em qualquer sistema operacional é o processo: uma abstração de um programa em execução. Tudo o mais depende desse conceito, e o projetista (e estudante) do sistema operacional deve ter uma compreensão profunda do que é um processo o mais cedo possível [...] Sem a abstração de processo, a computação moderna não poderia existir



Tanenbaum, Andrew S

- Processo é um programa em execução e que tem suas informações mantidas pelo S.O. - Unidade de trabalho dentro do sistema
  - Programa é uma entidade passiva, processo é uma entidade ativa.
- Além das informações sobre a execução, possui:
  - Recursos que o sistema pode utilizar;
  - Espaço de endereçamento;
  - Arquivos e outras Entradas/Saídas, etc.

- Multiprogramação: CPU muda de um processo para outro rapidamente, executando cada um por dezenas ou centenas de milissegundos;
- No curso de 1s ela pode trabalhar em vários deles, dando a ilusão do paralelismo (pseudoparalelismo)
  - Verdadeiro paralelismo: multiprocessadores.

- Multiprogramação: CPU muda de um processo para outro rapidamente, executando cada um por dezenas ou centenas de milissegundos;
- No curso de 1s ela pode trabalhar em vários deles, dando a ilusão do paralelismo (pseudoparalelismo)
  - Verdadeiro paralelismo: multiprocessadores.



- O sistema operacional deve ser capaz de:
  - Criar um processo
    - Inicialização do sistema;
    - Execução de uma chamada de sistema de criação de processo por um processo em execução;
    - Solicitação de um usuário para criar novo processo;
    - Início de uma tarefa em lote.
  - Reservar memória para um processo;
  - Organizar os processos para a espera do uso da CPU.

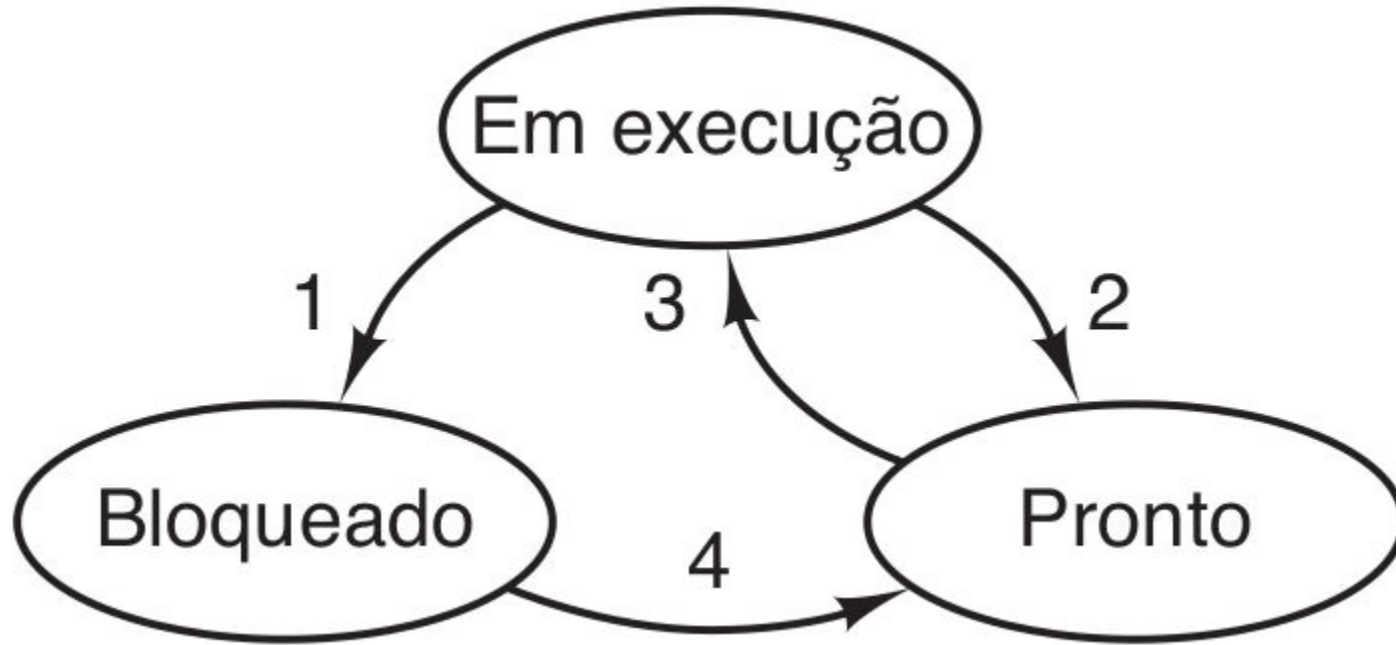
- Os vários processos criados pelo sistema operacional competem entre si pela atenção do processador;
- O gerenciador de processamento do sistema operacional é responsável por estabelecer a ordem de execução;
- Os processos podem estar em 3 estados
  - Em execução;
  - Pronto;
  - Bloqueado.

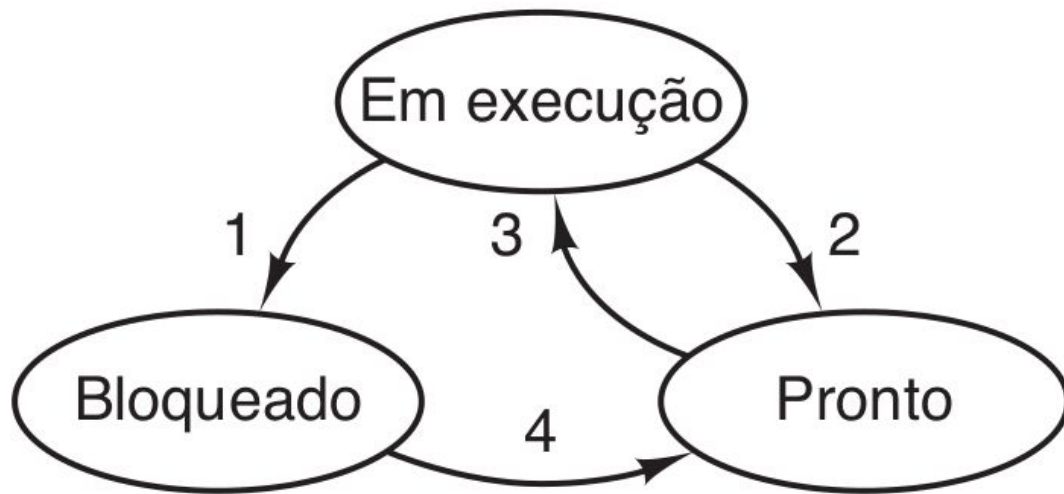
- Os vários processos criados pelo sistema operacional competem entre si pela atenção do processador;
- O gerenciador de processamento do sistema operacional é responsável por estabelecer a ordem de execução;
- Os processos podem estar em 3 estados
  - Em execução;
  - Pronto;
  - Bloqueado.





- Em execução
  - Quando o processo está, de fato, sendo processado pela CPU.
- Pronto
  - Quando o processo possui todas as condições necessárias para sua execução, porém, não está de posse da CPU;
  - Em geral, existem vários processos nesta fase, e o S.O. é responsável por selecionar o próximo processo a usar a CPU.
- Bloqueado
  - Quando o processo aguarda por algum evento externo ou por algum recurso do sistema indisponível no momento, por exemplo: informação de algum dispositivo de I/O, etc.





1. Processo bloqueado aguardando entrada
2. Escalonador seleciona outro processo
3. Escalonador seleciona o processo
4. Entrada torna-se disponível

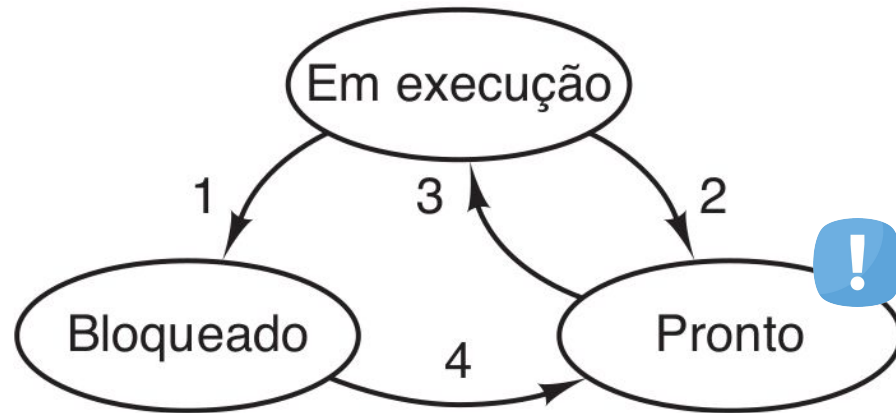
- Analisando processos na prática no sistema operacional:
  - GNU/Linux: comando ps, top;
  - Windows: gerenciador de tarefas;
  - Android: Dev Tools;
  - Etc.

- Término de processos
  - Voluntário
    - Saída normal;
    - Saída por erro.
  - Involuntário
    - Erro fatal;
    - Morto por outro processo.

- Como determinar qual processo utilizará a CPU quando estiver disponível?

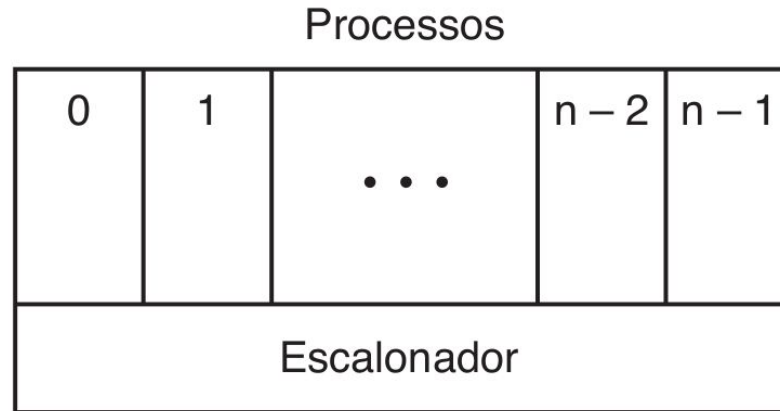
- Como determinar qual processo utilizará a CPU quando estiver disponível?
  - Escalonador de processos!

- Escalonador de processos
  - Age sempre sobre os processos em estado de PRONTO.





- Segue uma política de escolha – Algoritmo de escalonamento
  - Rotina é executada normalmente várias vezes por segundo.



- Objetivos das políticas de escalonamento:
  - Maximizar a utilização da CPU;
  - Maximizar o *throughput* (vazão);
  - Minimizar o turnaround (finalização do processo);
  - Minimizar o tempo de espera;
  - Minimizar o tempo de resposta;
  - Ser justa;
  - Maximizar o número de usuários interativos;
  - Ser previsível;
  - Minimizar e balancear o uso de recursos, etc.

- Objetivos das políticas de escalonamento:
  - Maximizar a utilização da CPU;
  - Maximizar o *throughput* (vazão);
  - Minimizar o turnaround (finalização do processo);
  - Minimizar o tempo de espera;
  - Minimizar o tempo de resposta;
  - Ser justa;
  - Maximizar o número de usuários interativos;
  - Ser previsível;
  - Minimizar e balancear o uso de recursos, etc.

- Objetivos das políticas de escalonamento:
  - Maximizar a utilização da CPU;
  - Maximizar o *throughput* (vazão);
  - Minimizar o turnaround (finalização do processo);
  - Minimizar o tempo de espera;
  - Minimizar o tempo de resposta;
  - Ser justa;
  - Maximizar o número de usuários interativos;
  - Ser previsível;
  - Minimizar e balancear o uso de recursos, etc.

- Objetivos das políticas de escalonamento:
  - Maximizar a utilização da CPU;
  - Maximizar o *throughput* (vazão);
  - Minimizar o turnaround (finalização do processo);
  - Minimizar o tempo de espera;
  - Minimizar o tempo de resposta;
  - Ser justa;
  - Maximizar o número de usuários interativos;
  - Ser previsível;
  - Minimizar e balancear o uso de recursos, etc.

- Objetivos das políticas de escalonamento:
  - Maximizar a utilização da CPU;
  - Maximizar o *throughput* (vazão);
  - Minimizar o turnaround (finalização do processo);
  - Minimizar o tempo de espera;
  - Minimizar o tempo de resposta;
  - Ser justa;
  - Maximizar o número de usuários interativos;
  - Ser previsível;
  - Minimizar e balancear o uso de recursos, etc.

- Objetivos das políticas de escalonamento:
  - Maximizar a utilização da CPU;
  - Maximizar o *throughput* (vazão);
  - Minimizar o turnaround (finalização do processo);
  - Minimizar o tempo de espera;
  - Minimizar o tempo de resposta;
  - Ser justa;
  - Maximizar o número de usuários interativos;
  - Ser previsível;
  - Minimizar e balancear o uso de recursos, etc.

- Objetivos das políticas de escalonamento:
  - Maximizar a utilização da CPU;
  - Maximizar o *throughput* (vazão);
  - Minimizar o turnaround (finalização do processo);
  - Minimizar o tempo de espera;
  - Minimizar o tempo de resposta;
  - Ser justa;
  - Maximizar o número de usuários interativos;
  - Ser previsível;
  - Minimizar e balancear o uso de recursos, etc.



- Objetivos das políticas de escalonamento:
  - Maximizar a utilização da CPU;
  - Maximizar o *throughput* (vazão);
  - Minimizar o turnaround (finalização do processo);
  - Minimizar o tempo de espera;
  - Minimizar o tempo de resposta;
  - Ser justa;
  - Maximizar o número de usuários interativos;
  - Ser previsível;
  - Minimizar e balancear o uso de recursos, etc.

- Objetivos das políticas de escalonamento:
  - Maximizar a utilização da CPU;
  - Maximizar o *throughput* (vazão);
  - Minimizar o turnaround (finalização do processo);
  - Minimizar o tempo de espera;
  - Minimizar o tempo de resposta;
  - Ser justa;
  - Maximizar o número de usuários interativos;
  - Ser previsível;
  - Minimizar e balancear o uso de recursos, etc.

- Objetivos das políticas de escalonamento:
  - Maximizar a utilização da CPU;
  - Maximizar o *throughput* (vazão);
  - Minimizar o turnaround (finalização do processo);
  - Minimizar o tempo de espera;
  - Minimizar o tempo de resposta;
  - Ser justa;
  - Maximizar o número de usuários interativos;
  - Ser previsível;
  - Minimizar e balancear o uso de recursos, etc.

- Objetivos das políticas de escalonamento:
  - Maximizar a utilização da CPU;
  - Maximizar o *throughput* (vazão);
  - Minimizar o turnaround (finalização do processo);
  - Minimizar o tempo de espera;
  - Minimizar o tempo de resposta;
  - Ser justa;
  - Maximizar o número de usuários interativos;
  - Ser previsível;
  - Minimizar e balancear o uso de recursos, etc.

- Como evitar que um processo monopolize o uso da CPU?

- Como evitar que um processo monopolize o uso da CPU?
  - Interrupção por tempo: QUANTUM ou TIME-SLICE;
  - Um processo interrompido por término do seu QUANTUM volta à fila em estado de PRONTO;
  - Um processo utiliza a CPU até:
    - Voluntariamente liberá-la;
    - QUANTUM expirado;
    - Outro processo exija a atenção da CPU (interrupção/prioridade);
    - Erro de execução.

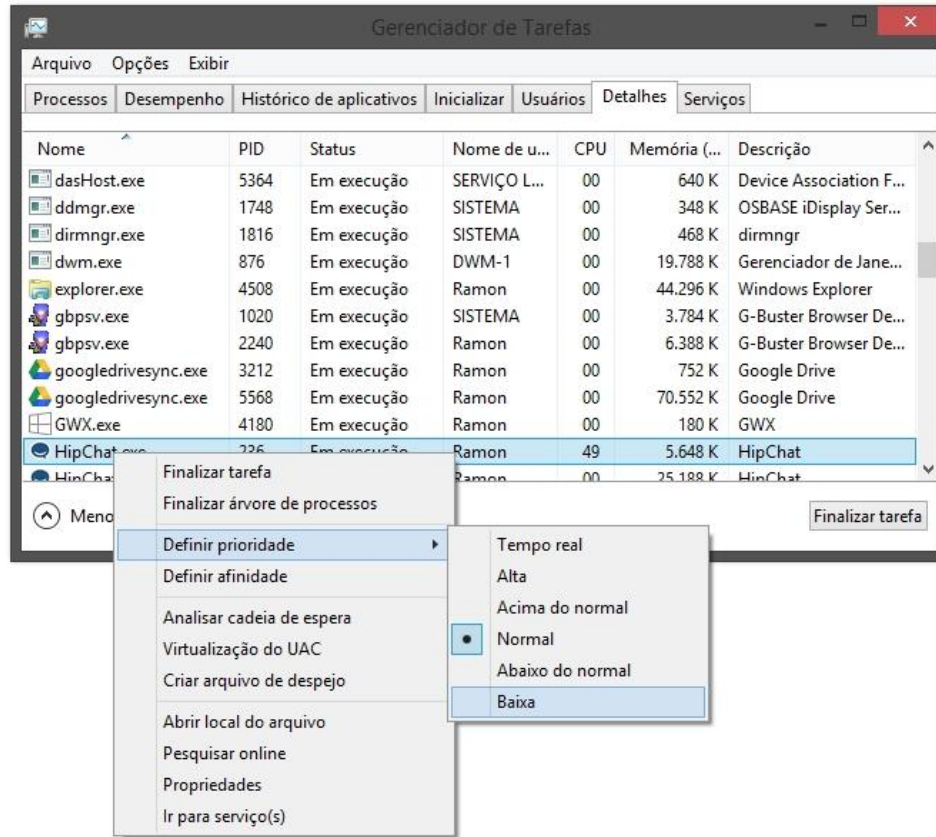
- Todos os processo têm a mesma importância?

- Todos os processo têm a mesma importância?
- Como diferenciá-los?



- Todos os processo têm a mesma importância?
- Como diferenciá-los?
  - Estabelecer PRIORIDADES!
    - Podem ser fixas ou mutáveis;
    - Podem ser associadas externamente ou de forma automatizada;
    - Podem ser definidas de forma racional ou arbitrária.

# Gerenciamento do Processador



# Gerenciamento do Processador

```
darth@darth-pc: ~ 150x38
top - 18:31:22 up 34 min, 1 user, load average: 0,28, 0,48, 0,45
Tasks: 197 total, 1 running, 196 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1,3 us, 1,0 sy, 0,0 ni, 97,5 id, 0,0 wa, 0,0 hi, 0,2 si, 0,0 st
MiB Mem : 5755,8 total, 2352,9 free, 1758,1 used, 1644,7 buff/cache
MiB Swap: 5940,0 total, 5940,0 free, 0,0 used. 3450,6 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR  S  %CPU  %MEM     TIME+ COMMAND
  855 root        20   0   589288 103532 72676 S   4,3   1,8   2:03.89 Xorg
 4716 darth      20   0   605824 500800 42208 S   3,0   0,8   0:00.36 lximage-qt
 1028 darth      20   0   669216 51844   38968 S   0,7   0,9   0:13.67 psensor
2096 darth      20   0 1301028 228464 124044 S   0,7   3,9   1:45.09 chrome
  503 root        20   0         0      0      0 I   0,3   0,0   0:02.00 kworker/3:3-events
  762 root        20   0    2500    772    704 S   0,3   0,0   0:05.11 acpid
1048 darth      20   0   747920 606800 43952 S   0,3   1,0   0:12.76 lxqt-panel
1049 darth      20   0   429188 35772   31736 S   0,3   0,6   0:00.34 lxqt-policykit-
2216 darth      20   0   376540 115176 56424 S   0,3   2,0   0:14.11 chrome
4179 darth      20   0   591112 111832 73184 S   0,3   1,9   0:06.21 chrome
4311 root        20   0         0      0      0 I   0,3   0,0   0:00.46 kworker/0:1-events
4346 root        20   0         0      0      0 I   0,3   0,0   0:00.73 kworker/2:1-events
4660 darth      20   0    11800   3536   3076 R   0,3   0,1   0:00.24 top
   1 root        20   0   165092 10368   7648 S   0,0   0,2   0:02.25 systemd
   2 root        20   0         0      0      0 S   0,0   0,0   0:00.00 kthreadd
   3 root         0 -20         0      0      0 I   0,0   0,0   0:00.00 rcu_gp
   4 root         0 -20         0      0      0 I   0,0   0,0   0:00.00 rcu_par_gp
   6 root         0 -20         0      0      0 I   0,0   0,0   0:00.45 kworker/0:0H-kblockd
   8 root         0 -20         0      0      0 I   0,0   0,0   0:00.00 mm_percpu_wq
   9 root        20   0         0      0      0 S   0,0   0,0   0:00.16 ksoftirqd/0
  10 root        20   0         0      0      0 I   0,0   0,0   0:01.42 rcu_sched
  11 root         rt   0         0      0      0 S   0,0   0,0   0:00.01 migration/0
  12 root       -51   0         0      0      0 S   0,0   0,0   0:00.00 idle_inject/0
  14 root        20   0         0      0      0 S   0,0   0,0   0:00.00 cpuhp/0
  15 root        20   0         0      0      0 S   0,0   0,0   0:00.00 cpuhp/1
  16 root       -51   0         0      0      0 S   0,0   0,0   0:00.00 idle_inject/1
  17 root         rt   0         0      0      0 S   0,0   0,0   0:00.01 migration/1
  18 root        20   0         0      0      0 S   0,0   0,0   0:00.16 ksoftirqd/1
  20 root         0 -20         0      0      0 I   0,0   0,0   0:00.00 kworker/1:0H-events_highpri
  21 root        20   0         0      0      0 S   0,0   0,0   0:00.00 cpuhp/2
  22 root       -51   0         0      0      0 S   0,0   0,0   0:00.00 idle_inject/2
```

# Gerenciamento do Processador

darth@darth-pc: ~ 150x38

top - 18:31:22 up 34 min, 1 user, load average: 0,28, 0,48, 0,45  
Tasks: 197 total, 1 running, 196 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 1,3 us, 1,0 sy, 0,0 ni, 97,5 id, 0,0 wa, 0,0 hi, 0,2 si, 0,0 st  
MiB Mem : 5755,8 total, 2252,3 free, 1758,1 used, 1644,7 buff/cache  
MiB Swap: 5940,0 total, 5940,0 free, 0,0 used. 3450,6 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	%MEM	TIME+	COMMAND
855	root	20	0	589288	103532	72676	S	4,3	1,8	2:03.89	Xorg
4716	darth	20	0	605824	50080	42208	S	3,0	0,8	0:00.36	lximage-qt
1028	darth	20	0	669216	51844	38968	S	0,7	0,9	0:13.67	psensor
2096	darth	20	0	301028	228464	124044	S	0,7	1,9	1:45.09	chrome
503	root	20	0	0	0	0	I	0,3	0,0	0:02.00	kworker/3:3-events
762	root	20	0	2500	772	704	S	0,3	0,0	0:05.11	acpid
1048	darth	20	0	747920	60680	43952	S	0,3	1,0	0:12.76	lxqt-panel
1049	darth	20	0	429188	35772	31736	S	0,3	0,6	0:00.34	lxqt-policykit-
2216	darth	20	0	376540	115176	56424	S	0,3	2,0	0:14.11	chrome
4179	darth	20	0	591112	111832	73184	S	0,3	1,9	0:06.21	chrome
4311	root	20	0	0	0	0	I	0,3	0,0	0:00.46	kworker/4:1-events
4346	root	20	0	0	0	0	I	0,3	0,0	0:00.73	kworker/2:1-events
4660	darth	20	0	11800	3536	3076	R	0,3	0,1	0:00.24	top
1	root	20	0	165092	10368	7648	S	0,0	0,2	0:02.25	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_par_gp
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.45	kworker/0:0H-kblockd
8	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	mm_percpu_wq
9	root	20	0	0	0	0	S	0,0	0,0	0:00.16	ksoftirqd/0
10	root	20	0	0	0	0	I	0,0	0,0	0:01.42	rcu_sched
11	root	rt	0	0	0	0	S	0,0	0,0	0:00.01	migration/0
12	root	-51	0	0	0	0	S	0,0	0,0	0:00.00	idle_inject/0
14	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/0
15	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/1
16	root	-51	0	0	0	0	S	0,0	0,0	0:00.00	idle_inject/1
17	root	rt	0	0	0	0	S	0,0	0,0	0:00.01	migration/1
18	root	20	0	0	0	0	S	0,0	0,0	0:00.16	ksoftirqd/1
20	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/1:0H-events_highpri
21	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/2
22	root	-51	0	0	0	0	S	0,0	0,0	0:00.00	idle_inject/2

- 20 a +19
- Quanto menor, maior a prioridade

- Um processo usa a CPU até seu fim?
- É possível um evento externo causar a perda do uso da CPU de um processo?

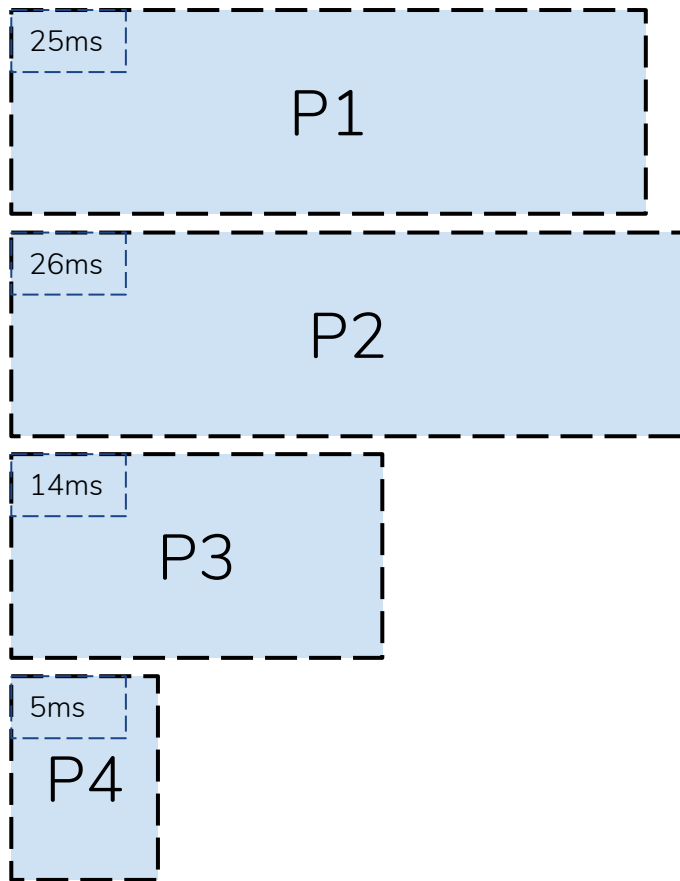
- Um processo usa a CPU até seu fim?
- É possível um evento externo causar a perda do uso da CPU de um processo?
  - PREEMPCÃO
    - Não preemptivo
      - Não pode ser interrompido externamente.
    - Preemptivo
      - Pode interromper a execução de um processo.

# Algoritmos de escalonamento

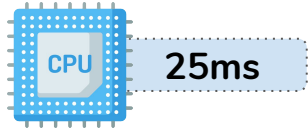
- FIFO;
- SJF (*Shortest Job First*);
- Round-Robin;
- SRT (*Shortest Remaining Time*);
- Prioridade;
- Múltiplas filas;
- Múltiplas filas com realimentação;
- Escalonamento para vários processadores;
- Escalonamento de tempo real.



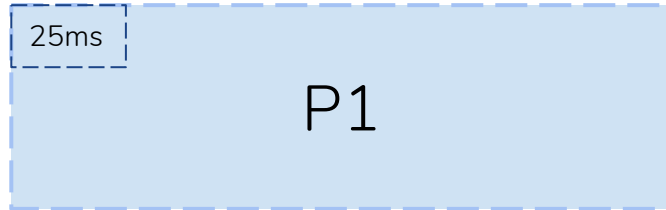
- FIFO - First in, first out
  - Não preemptivo;
  - O processo que chega primeiro é o primeiro a ser escolhido para usar a CPU (FIFO);
  - Fila de processos;
  - Criado originalmente para sistemas batch;
  - Injusto para processos curtos;
  - Ineficiente para sistemas de tempo compartilhado.



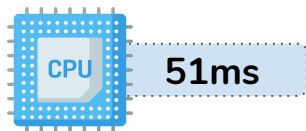
FIFO



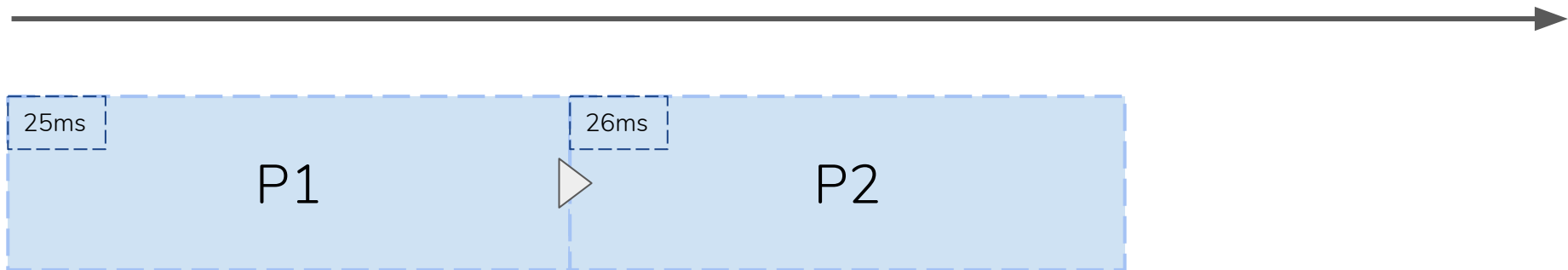
P1: 25ms  
P2: 26ms  
P3: 14ms  
P4: 5ms



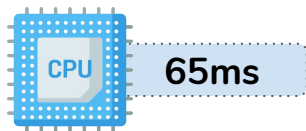
FIFO



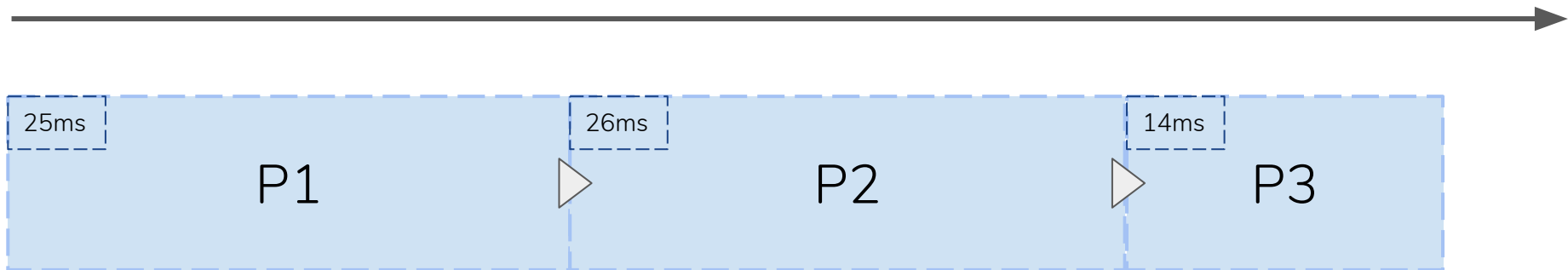
P1: 25ms  
P2: 26ms  
P3: 14ms  
P4: 5ms



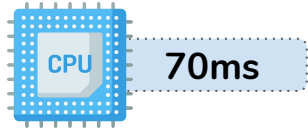
FIFO



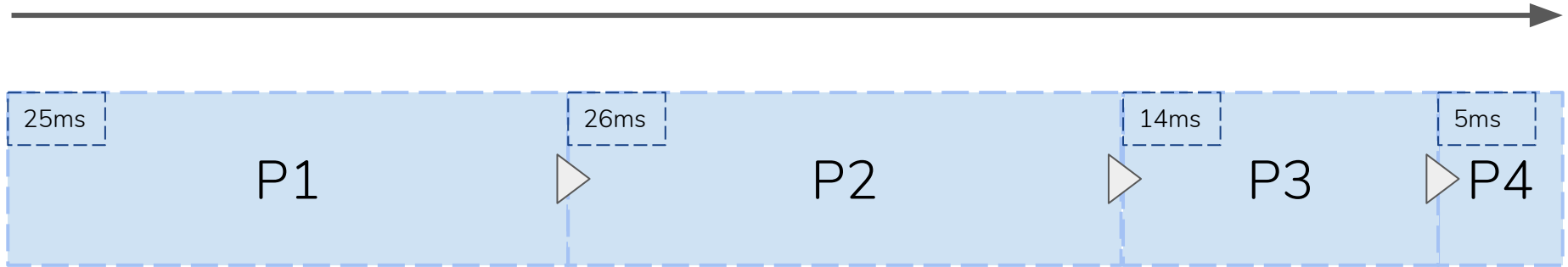
P1: 25ms  
P2: 26ms  
P3: 14ms  
P4: 5ms



FIFO

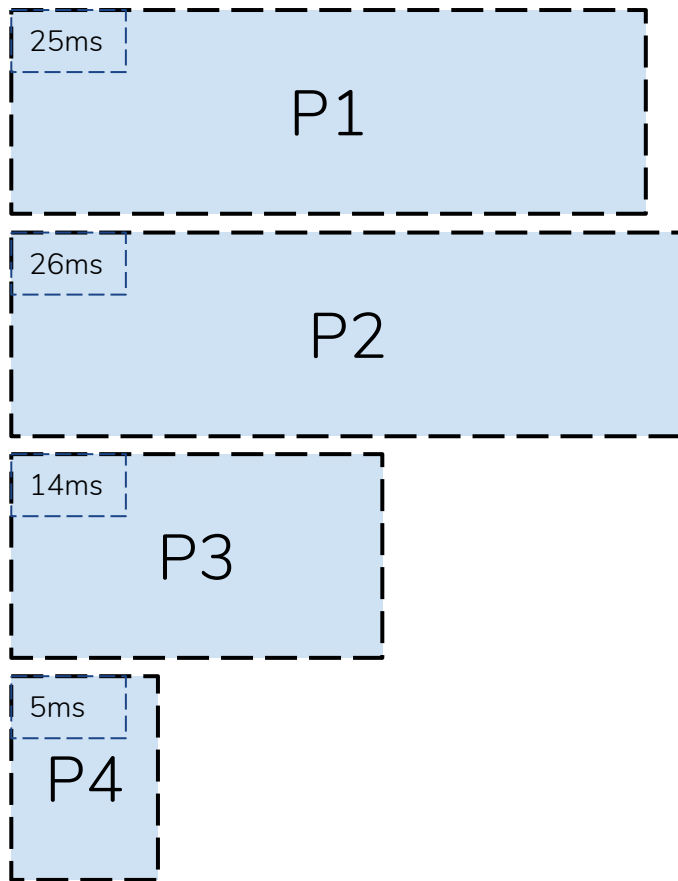


P1: 25ms  
P2: 26ms  
P3: 14ms  
P4: 5ms



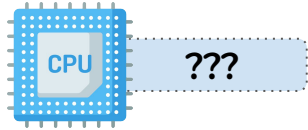
Tempo total de execução: 70ms

- SJF - Shortest Job First
  - Não preemptivo;
  - Processo com menor tempo para ser completado é escolhido;
  - Se tempo de execução não estiver disponível: Estimativa
    - Desempate pode ser feito por FIFO.
  - Reduz o tempo de espera, com o mínimo para um conjunto de processos;
  - Problemas: grande variância no tempo de espera, necessidade de “prever o futuro”.





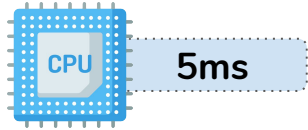
SJF



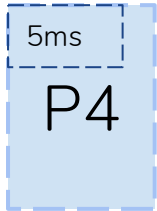
P1: 25ms  
P2: 26ms  
P3: 14ms  
P4: 5ms



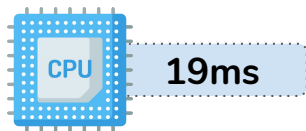
SJF



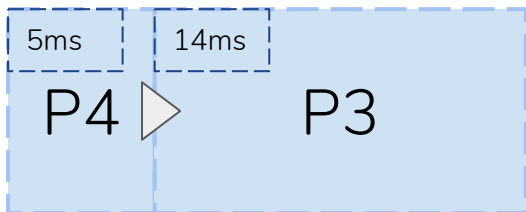
P1: 25ms  
P2: 26ms  
P3: 14ms  
**P4: 5ms**



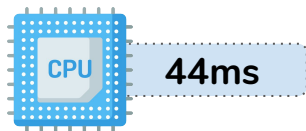
SJF



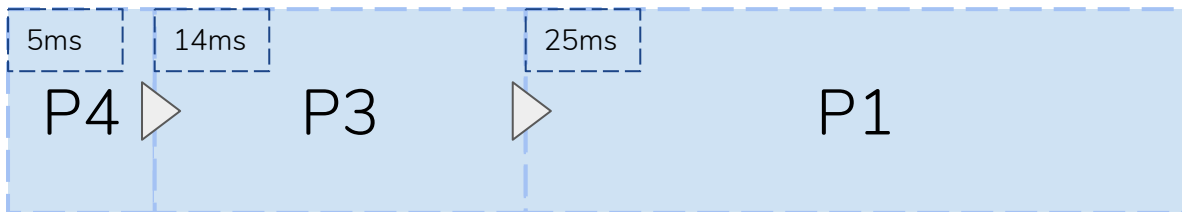
P1: 25ms  
P2: 26ms  
**P3: 14ms**  
**P4: 5ms**



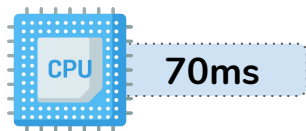
SJF



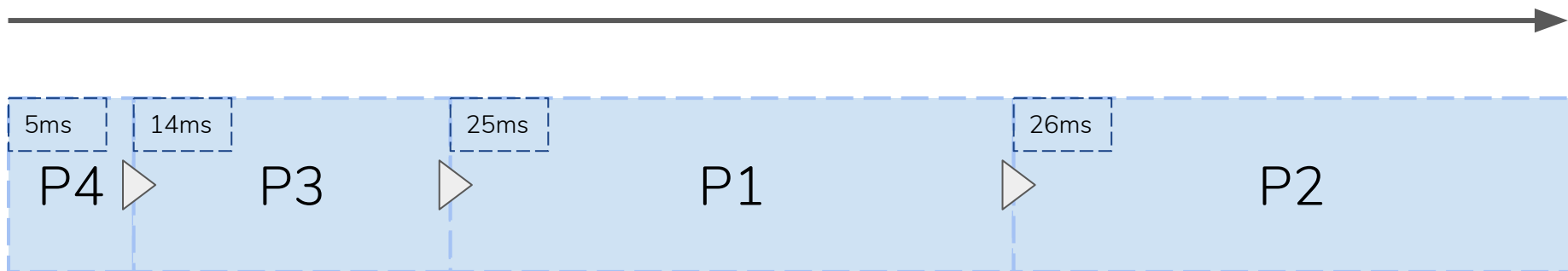
P1: 25ms  
P2: 26ms  
P3: 14ms  
P4: 5ms



SJF



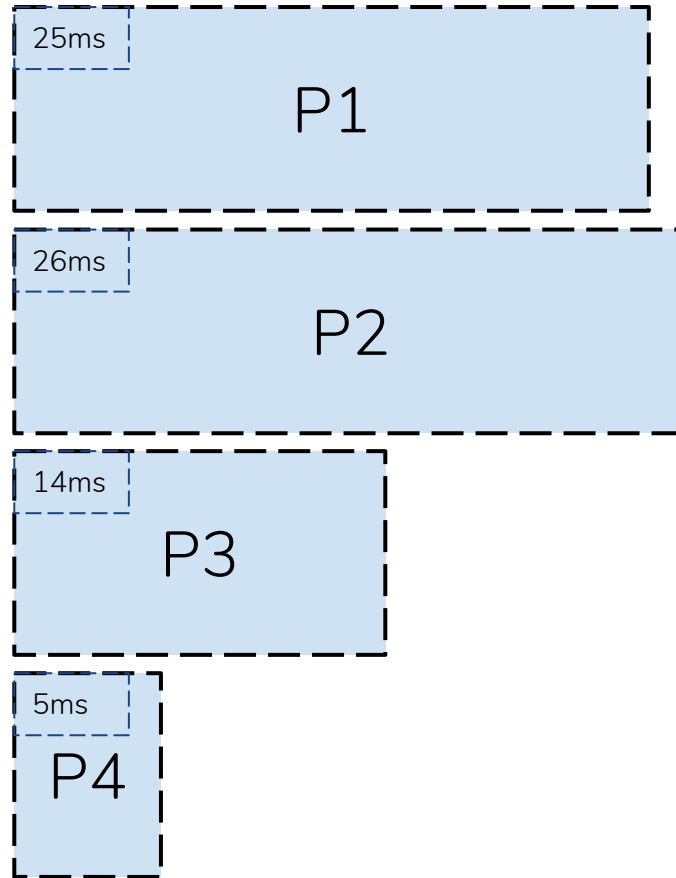
P1: 25ms  
P2: 26ms  
P3: 14ms  
P4: 5ms



Tempo total de execução: 70ms

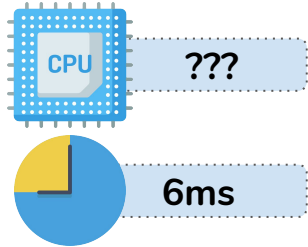
- Round Robin
  - Um dos mais antigos algoritmos de escalonamento;
  - Simples, justos e amplamente utilizado
    - DNS, balanceamento de carga, etc.
  - A cada processo é designado um intervalo
    - QUANTUM.
  - Preemptivo.

- Após o QUANTUM o processo sofre PREEMPÇÃO
- Fila de PRONTO é tratada como uma FIFO circular;
- Escalonador seleciona primeiro processo da fila de PRONTO para ser executado por 1 QUANTUM;
  - Se o tempo necessário para o processo for maior que 1 QUANTUM é feito o CHAVEAMENTO DE CONTEXTO
    - Salva estado, vai para status de PRONTO, seleciona novo processo.



Quantum: 6ms

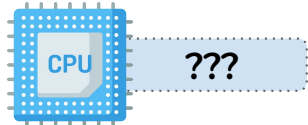




## Round Robin

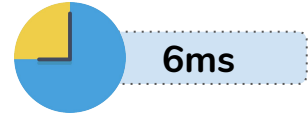
P1: 25ms  
P2: 26ms  
P3: 14ms  
P4: 5ms





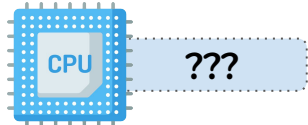
## Round Robin

P1: 19ms  
P2: 26ms  
P3: 14ms  
P4: 5ms



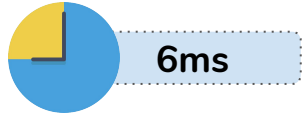
6ms

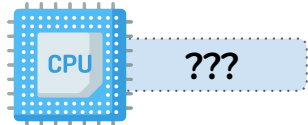
P1



## Round Robin

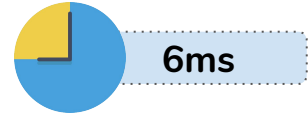
P1: 19ms  
P2: 20ms  
P3: 14ms  
P4: 5ms



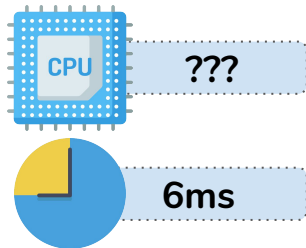


## Round Robin

P1: 19ms  
P2: 20ms  
P3: 8ms  
P4: 5ms



# Gerenciamento do Processador > Algoritmos escalonamento

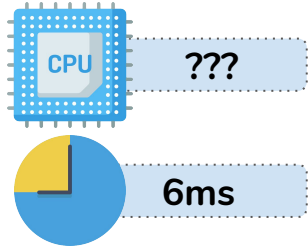


Round Robin

P1: 19ms  
P2: 20ms  
P3: 8ms  
**P4: 0ms**



# Gerenciamento do Processador > Algoritmos escalonamento

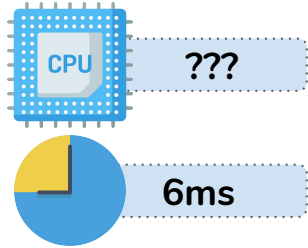


Round Robin

P1: 19ms  
P2: 20ms  
P3: 8ms  
**P4: 0ms**

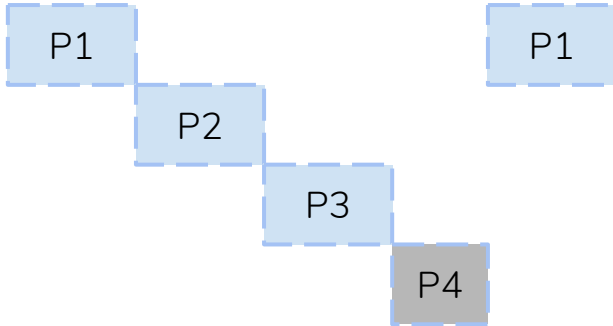


# Gerenciamento do Processador > Algoritmos escalonamento

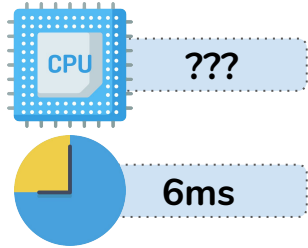


Round Robin

P1: 13ms  
P2: 20ms  
P3: 8ms  
**P4: 0ms**

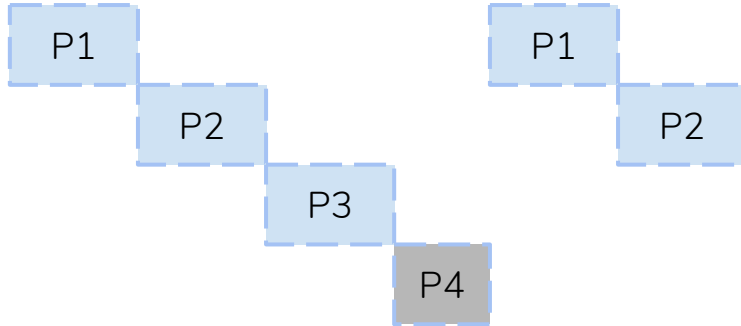


# Gerenciamento do Processador > Algoritmos escalonamento



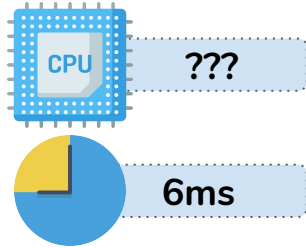
Round Robin

P1: 13ms  
P2: 14ms  
P3: 8ms  
**P4: 0ms**



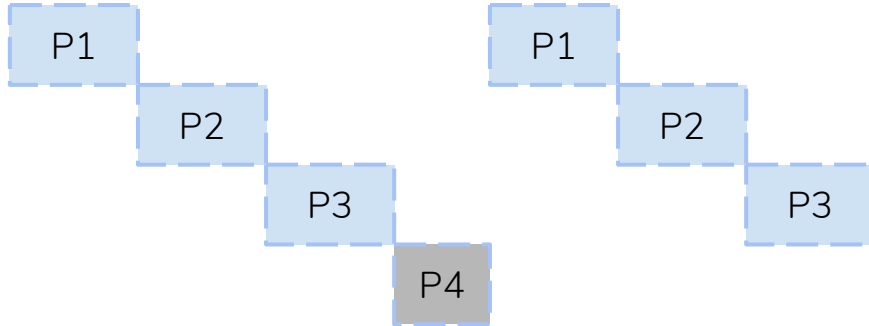


# Gerenciamento do Processador > Algoritmos escalonamento

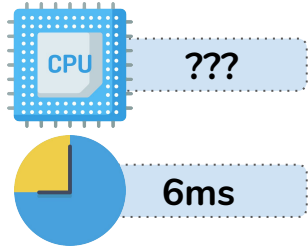


Round Robin

P1: 13ms  
P2: 14ms  
P3: 2ms  
**P4: 0ms**

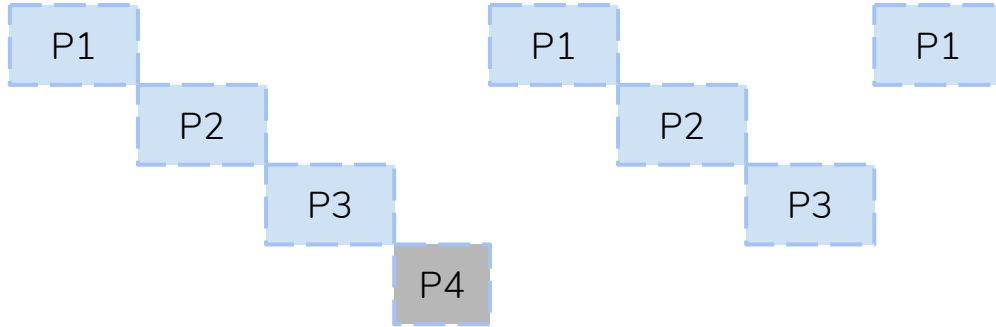


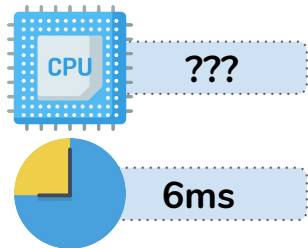
# Gerenciamento do Processador > Algoritmos escalonamento



## Round Robin

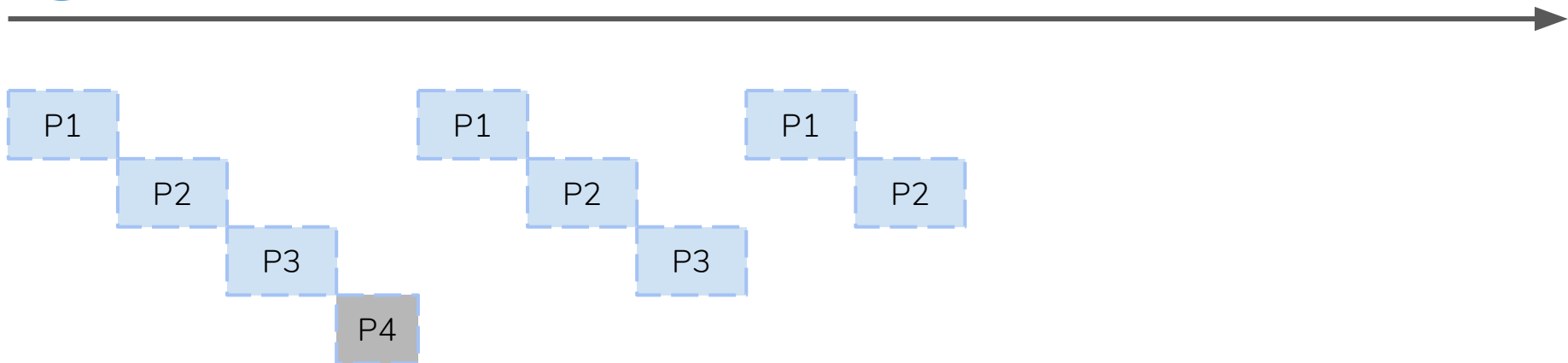
P1: 7ms  
P2: 14ms  
P3: 2ms  
**P4: 0ms**



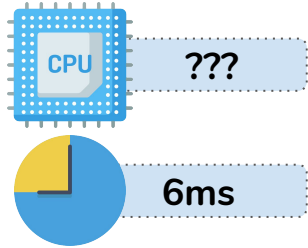


## Round Robin

P1: 7ms  
P2: 8ms  
P3: 2ms  
**P4: 0ms**

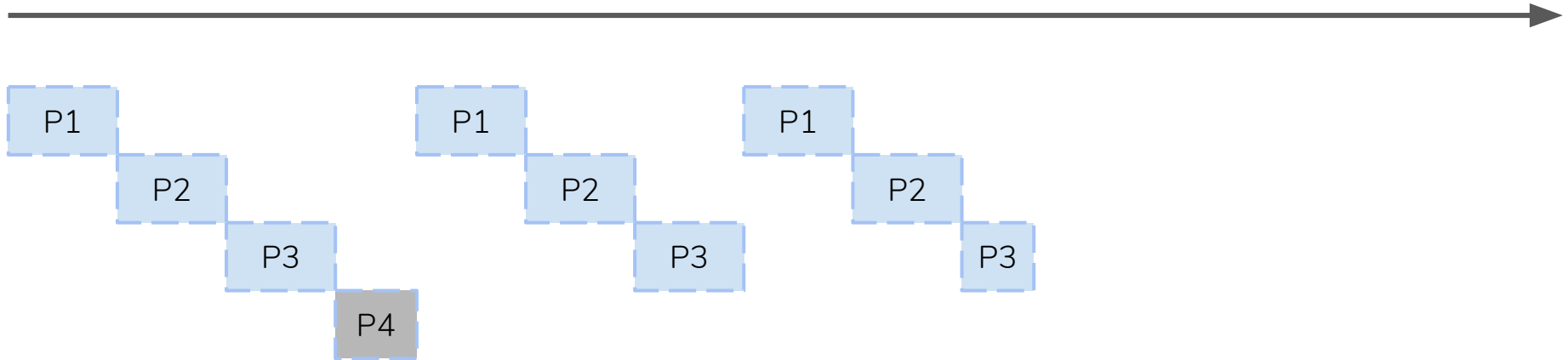


# Gerenciamento do Processador > Algoritmos escalonamento

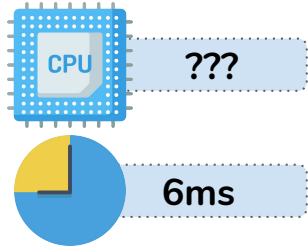


Round Robin

P1: 7ms  
P2: 8ms  
**P3: 0ms**  
**P4: 0ms**

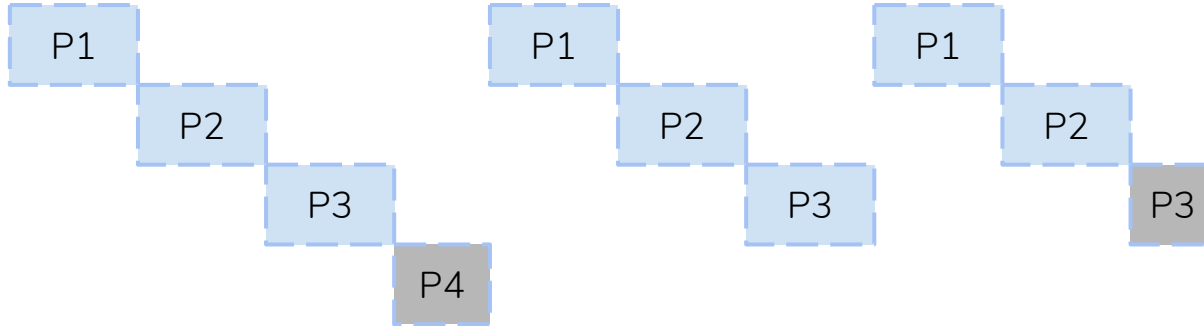


# Gerenciamento do Processador > Algoritmos escalonamento

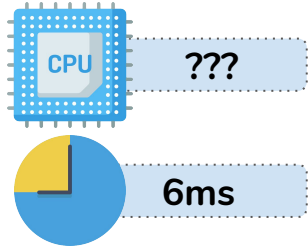


Round Robin

P1: 7ms  
P2: 8ms  
**P3: 0ms**  
**P4: 0ms**

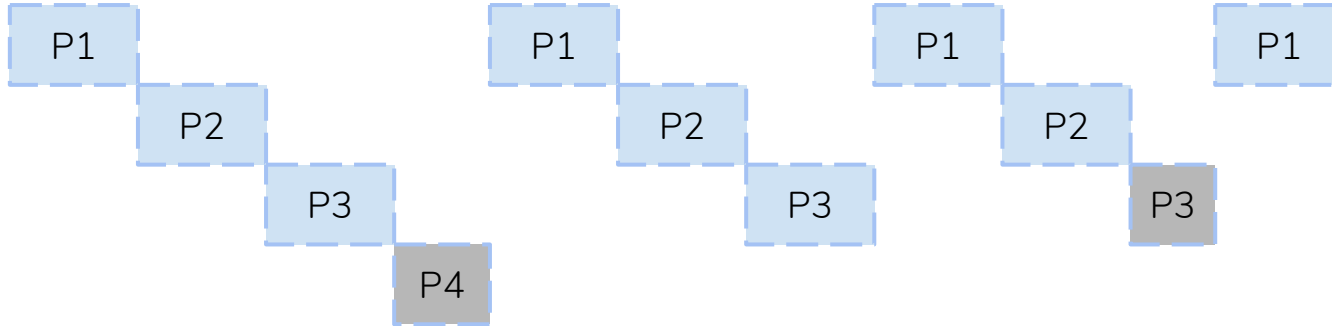


# Gerenciamento do Processador > Algoritmos escalonamento

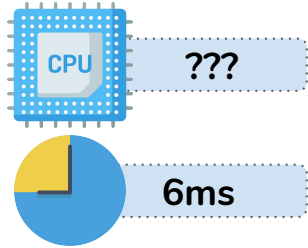


## Round Robin

P1: 1ms  
P2: 8ms  
**P3: 0ms**  
**P4: 0ms**

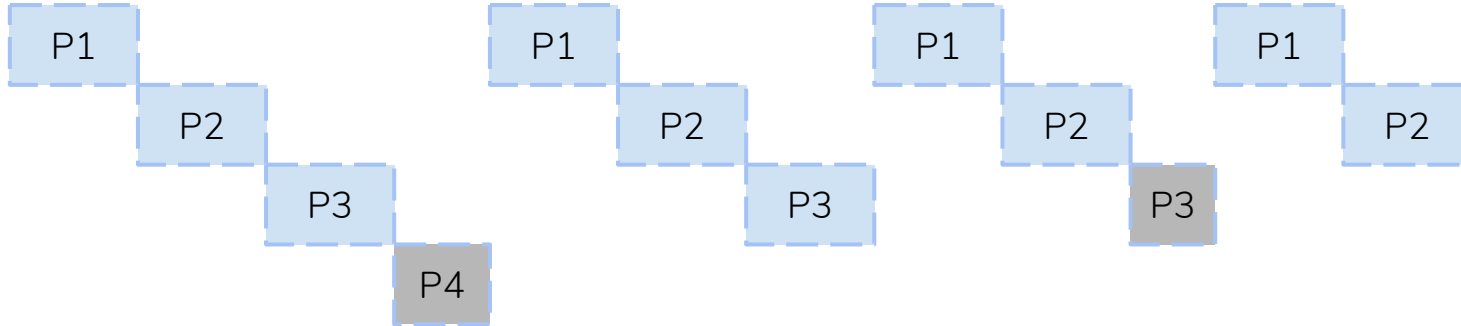


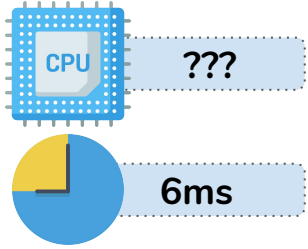
# Gerenciamento do Processador > Algoritmos escalonamento



## Round Robin

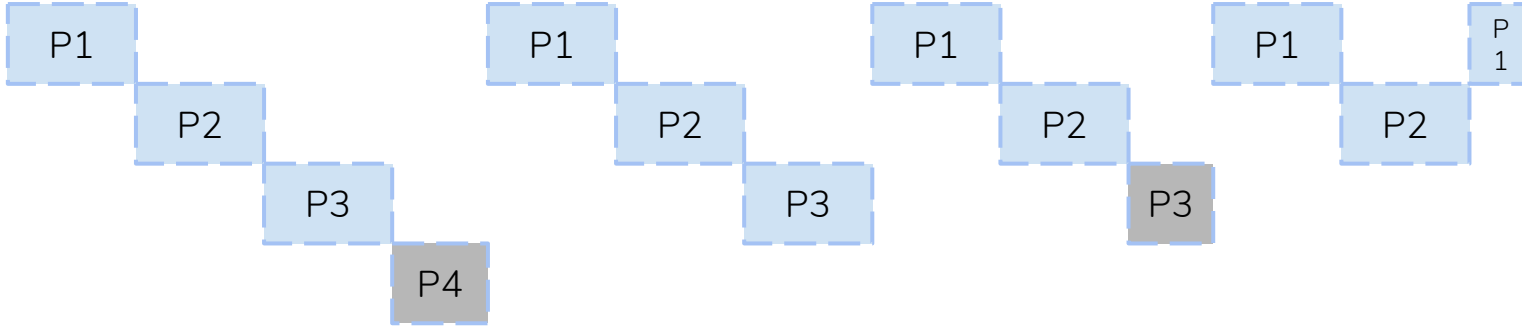
P1: 1ms  
P2: 2ms  
**P3: 0ms**  
**P4: 0ms**





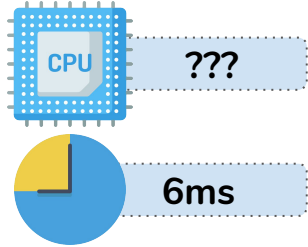
## Round Robin

**P1: 0ms**  
P2: 2ms  
**P3: 0ms**  
**P4: 0ms**



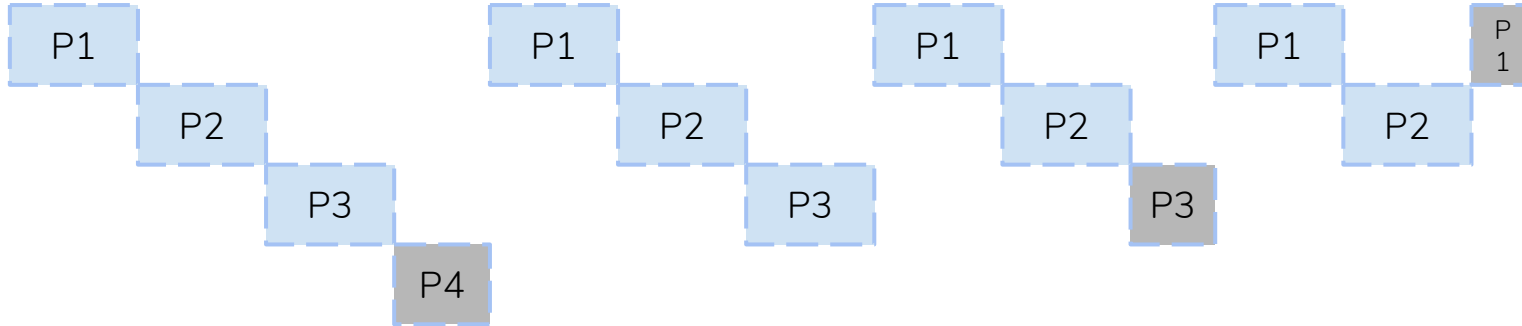


# Gerenciamento do Processador > Algoritmos escalonamento

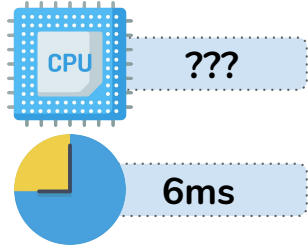


## Round Robin

P1: 0ms  
P2: 2ms  
P3: 0ms  
P4: 0ms

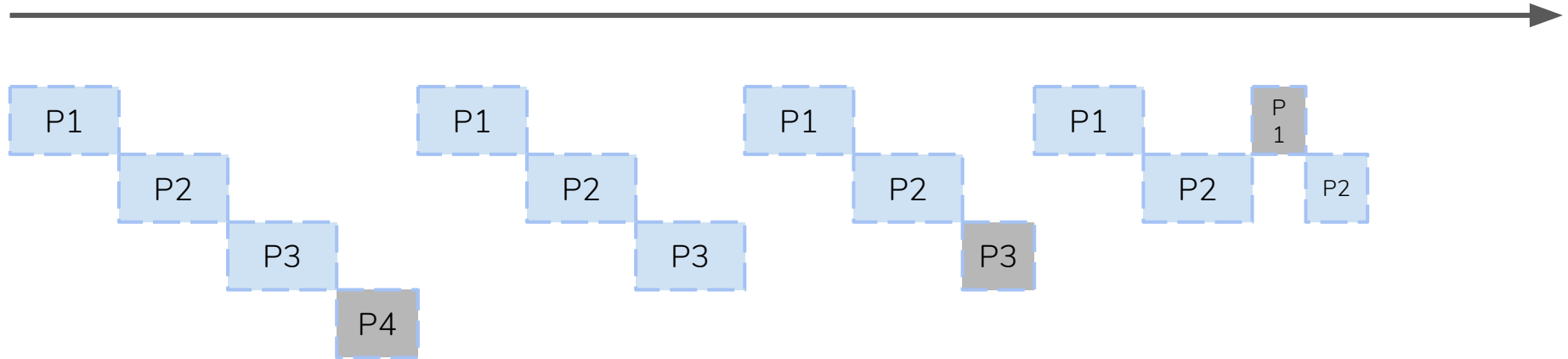


# Gerenciamento do Processador > Algoritmos escalonamento

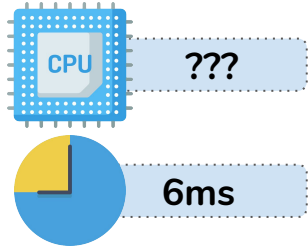


## Round Robin

P1: 0ms  
P2: 0ms  
P3: 0ms  
P4: 0ms

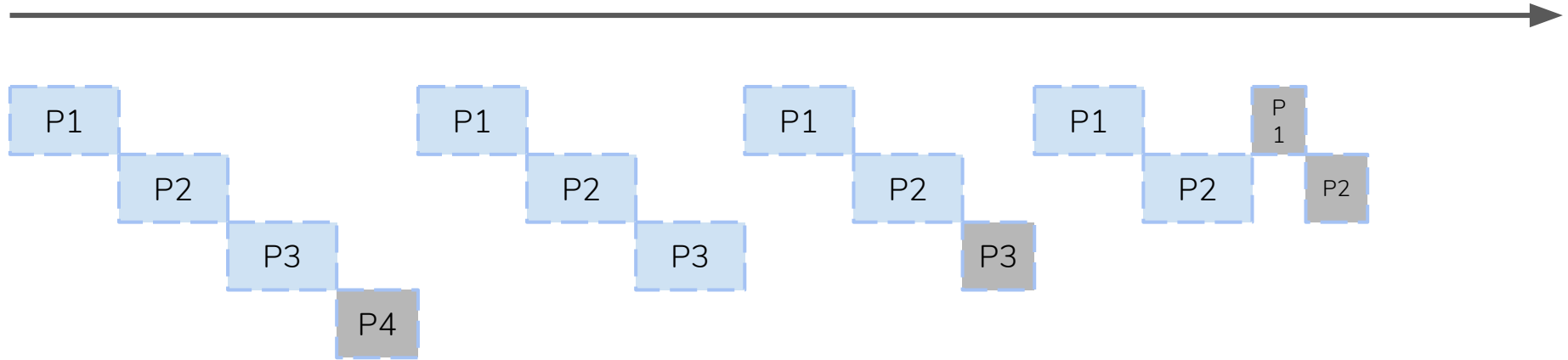


# Gerenciamento do Processador > Algoritmos escalonamento



## Round Robin

P1: 0ms  
P2: 0ms  
P3: 0ms  
P4: 0ms



- SRT - Shortest Remaining Time
  - “SJF Preemptivo”;

- SRT - Shortest Remaining Time
  - “SJF Preemptivo”;
  - É escolhido o processo com menor tempo para ser completado;
  - Um processo em execução é interrompido quando um novo processo com menor tempo para ser completado aparece na fila com o status de PRONTO.



- Escalonamento por prioridade
  - Cada processo tem uma prioridade associada;
  - CPU é alocada para maior prioridade;
  - SJF pode ser considerado um caso de escalonamento por prioridade



- Escalonamento por prioridade
  - Cada processo tem uma prioridade associada;
  - CPU é alocada para maior prioridade;
  - SJF pode ser considerado um caso de escalonamento por prioridade
    - Considera-se o tempo como priorização;
  - Geralmente a prioridade é representada por um número.



- Prioridade definida:
  - Internamente ou externamente.
- Problema: *Starvation* – Processos de baixa prioridade podem ficar indefinidamente em estado de PRONTO;
- Solução: aumentar progressivamente a prioridade dos processos em estado de PRONTO aguardando a vez de usar a CPU.



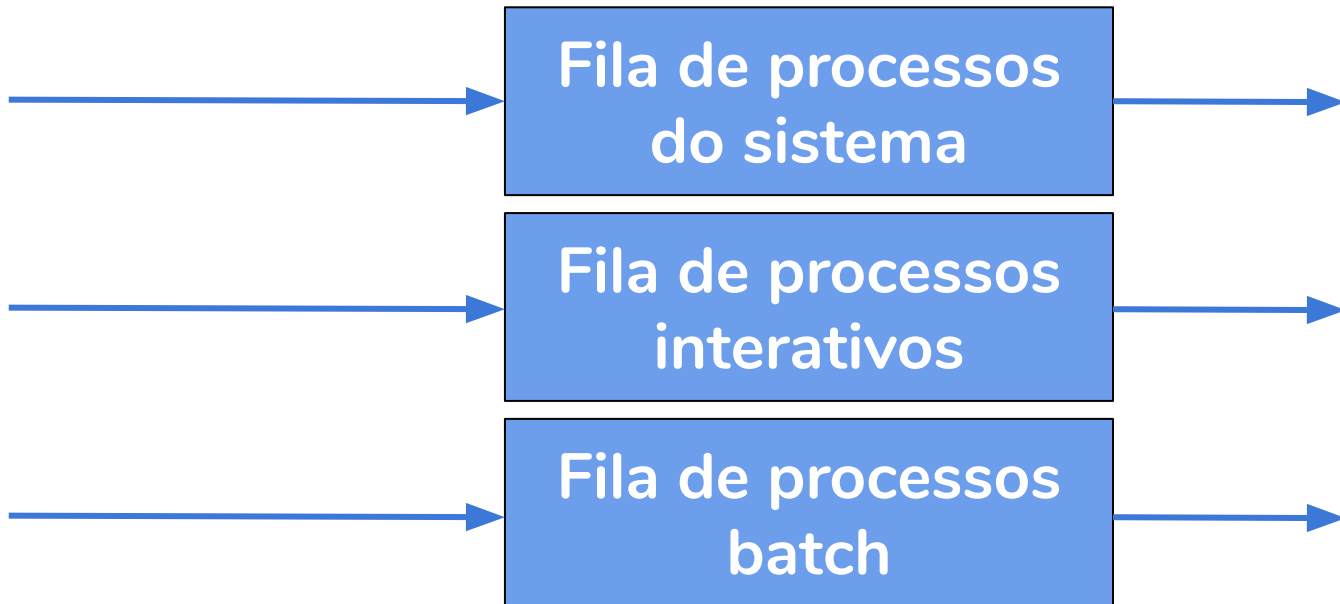
<https://scopeweb.mit.edu>

## Starvation:

Quando os responsáveis foram desligar o IBM 7094 no MIT em 1973, descobriram um processo com baixa prioridade que estava em estado de PRONTO desde 1967 e ainda não havia sido processado.

- Múltiplas filas
  - Processos divididos em grupos
    - Em função do processamento
      - Sistema;
      - Interativo;
      - Batch.
    - Cada um com mecanismos de escalonamento
      - Interativo - Round Robin;
      - Batch - FIFO.

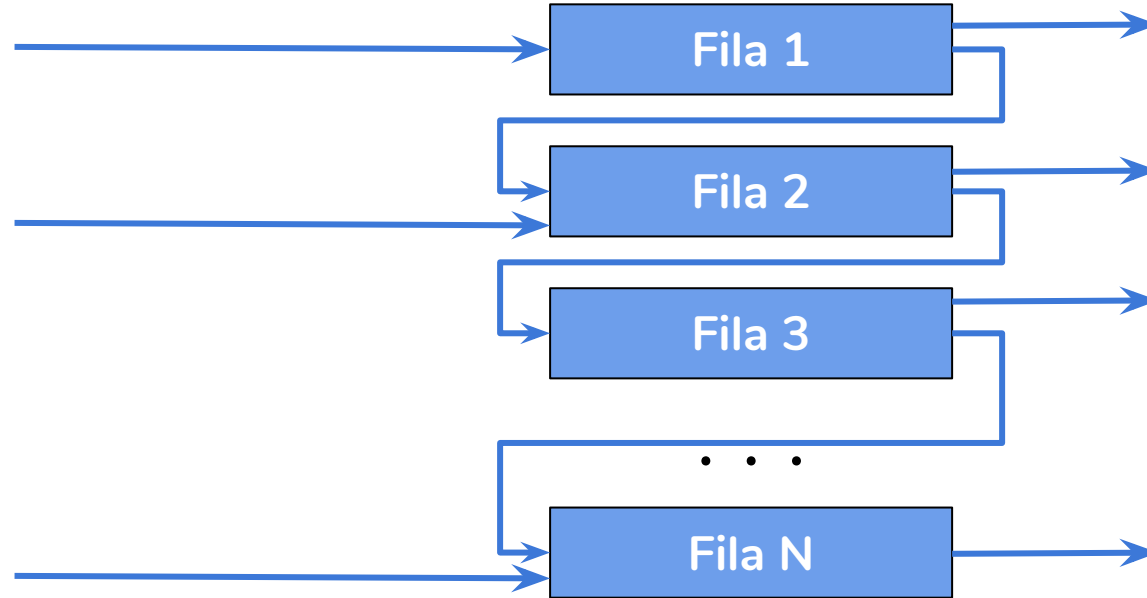
Alta prioridade



Baixa prioridade

- Múltiplas filas com realimentação
  - Processos não permanecem na mesma fila até o fim do processamento;
  - S.O. faz um ajuste dinâmico;
  - Parâmetros para escalonamento
    - Número de filas;
    - Algoritmo para cada fila;
    - Método para alterar fila de um processo, etc.
  - Método mais complexo.

Alta prioridade



Baixa prioridade

- Escalonamento para vários processadores
  - Mais complexo;
  - Processadores idênticos
  - Compartilhamento de carga;
  - Fila por processador;
  - Fila única: compartilhamento de dados.

- Abordagens do algoritmo
  - Simétrico: Cada processador faz seu próprio escalonamento;
  - Mestre-escravo (assimétrico): Um algoritmo é executado em um processador reservado somente para esse fim, e escalona os outros processos em outros processadores.



- Sistemas de tempo real
  - Hard Real Time: Requisitos rígidos, tempo é essencial
    - Uma tarefa crítica deve ser completada dentro de um tempo GARANTIDO
    - Avião, sistemas médicos, indústria, etc.
  - Soft Real Time: Requisitos flexíveis
    - Processo crítico recebe prioridade sobre os outros com menor importância
    - Realidade aumentada, multimídia, etc.
- Sistemas de Real Time rígidos requerem processadores dedicados.



<https://news.sanfordhealth.org>



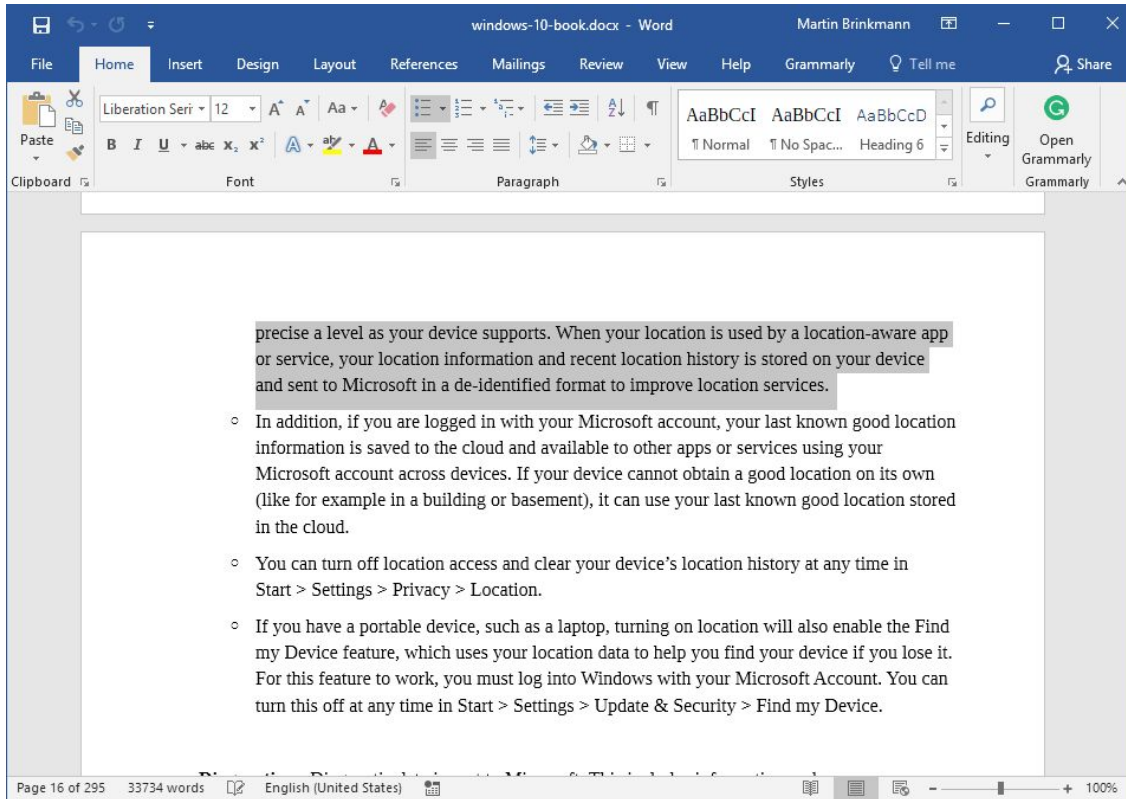
<https://www.wired.com/story/boeing-737-max-8-ethiopia-crash-faa-software-fix-lion-air/>

# Threads

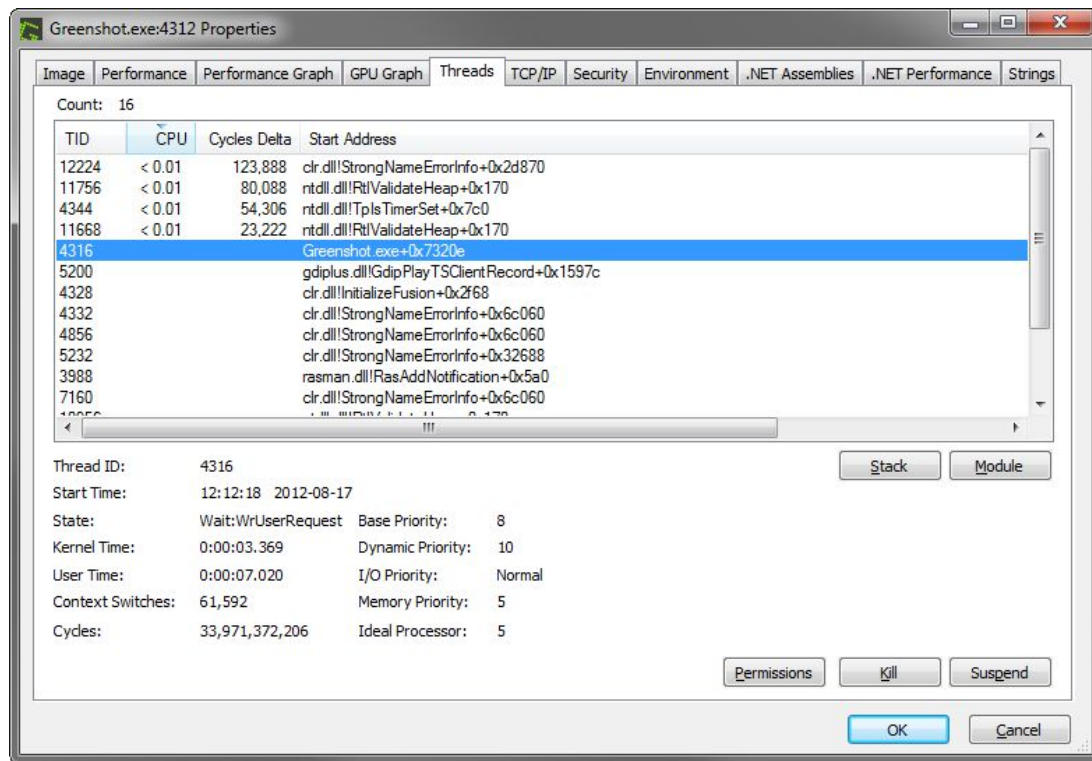
- Cada processo tem um espaço de endereçamento e um único *thread* de controle;
- Em muitas situações é desejável ter múltiplos *threads* de controle no mesmo espaço de endereçamento
  - Executando quase em paralelo;
  - Como se fossem processos separados
    - Exceto pelo espaço de endereçamento compartilhado.

- Uma forma de um processo dividir a si mesmo;
- Executam “dentro” de um processo;
- Conhecidas como “processos leves”;
- Possibilidade de associar mais de um fluxo de execução
  - Compartilhamento de recursos
    - Espaço de endereçamento
  - Comunicação entre eles facilitada;
  - Chaveamento de contexto total desnecessário.

Na prática...



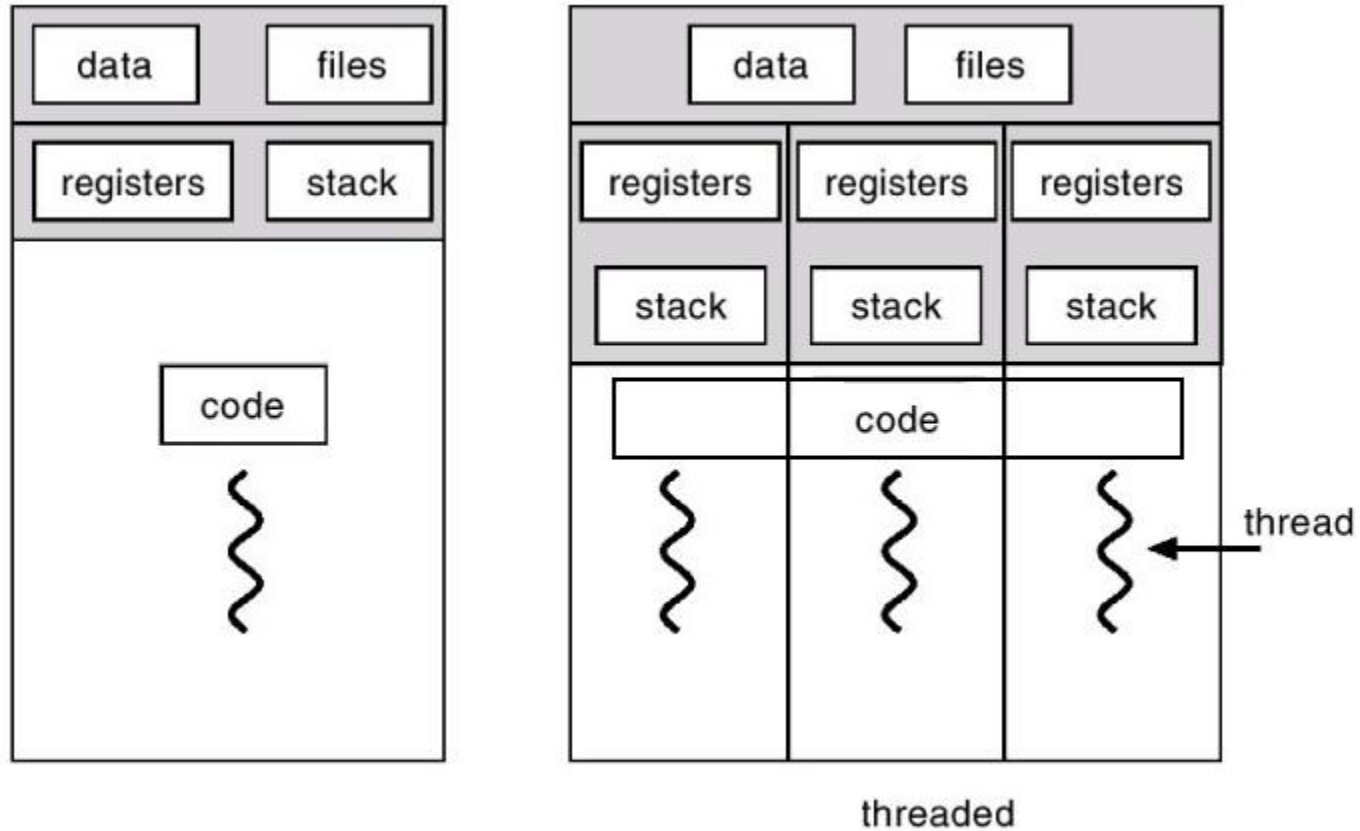
- Windows: Process Explorer





- Linux
  - `top -H`
  - `htop`
  - `ps -eT`

- Processos x *Threads*
  - Processo: programa em execução que contém um único fluxo de execução;
  - *Threads*: programa em execução com múltiplos fluxos de execução.



- Se um processo tem várias *Threads*, elas compartilham quase todos os recursos e memória;
- Cada *Thread* tem sua própria pilha de execução e registradores;
- Em um processo com múltiplas *Threads*, quando uma *Thread* está bloqueada aguardando, uma segunda *Thread* no mesmo processo pode executar
- Aplicações que requerem compartilhamento de dados se beneficiam ao usar *Threads*.

- Benefícios:
  - Velocidade de criação das *Threads*;
  - Capacidade de resposta;
  - Compartilhamento de recursos;
  - Economia de recursos;
  - Desempenho.

- Principais diferenças
  - Processos são geralmente independentes, enquanto Threads são subconjuntos de um processo;
  - Processos carregam mais informações que Threads;
  - Processos tem espaço de endereçamento separado;
  - Processos interagem somente através de mecanismos de comunicação entre processos fornecidos pelo S.O.;
  - Troca de contexto entre Threads usa o mesmo procedimento que nos processos, porém é tipicamente mais rápida.

- Como implementar *Threads* na prática?



- Como implementar *Threads* na prática?



## The Java™ Tutorials

[Hide TOC](#)

### Concurrency

#### Processes and Threads

#### Thread Objects

##### Defining and Starting a Thread

##### Pausing Execution with Sleep

##### Interrupts

##### Joins

##### The SimpleThreads Example

#### Synchronization

##### Thread Interference

##### Memory Consistency

##### Errors

##### Synchronized Methods

[« Previous](#) • [Trail](#) • [Next »](#)[Home Page](#) > [Essential Classes](#) > [Concurrency](#)

*The Java Tutorials have been written for JDK 8. Examples and practices described in this page don't take advantage of improvements introduced in later releases.*

## Thread Objects

Each thread is associated with an instance of the class `Thread`. There are two basic strategies for using `Thread` objects to create a concurrent application.

- To directly control thread creation and management, simply instantiate `Thread` each time the application needs to initiate an asynchronous task.
- To abstract thread management from the rest of your application, pass the application's tasks to an *executor*.

This section documents the use of `Thread` objects. Executors are discussed with other [high-level concurrency objects](#).

[« Previous](#) • [Trail](#) • [Next »](#)



- Como implementar *Threads* na prática?

```
class PrimeThread extends Thread {  
    long minPrime;  
    PrimeThread(long minPrime) {  
        this.minPrime = minPrime;  
    }  
  
    public void run() {  
        // compute primes larger than minPrime  
        . . .  
    }  
}
```

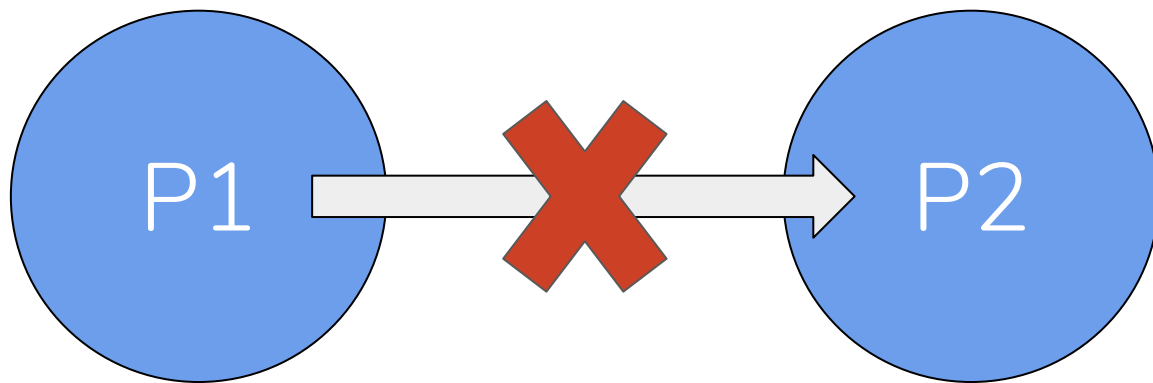
- Como implementar *Threads* na prática?



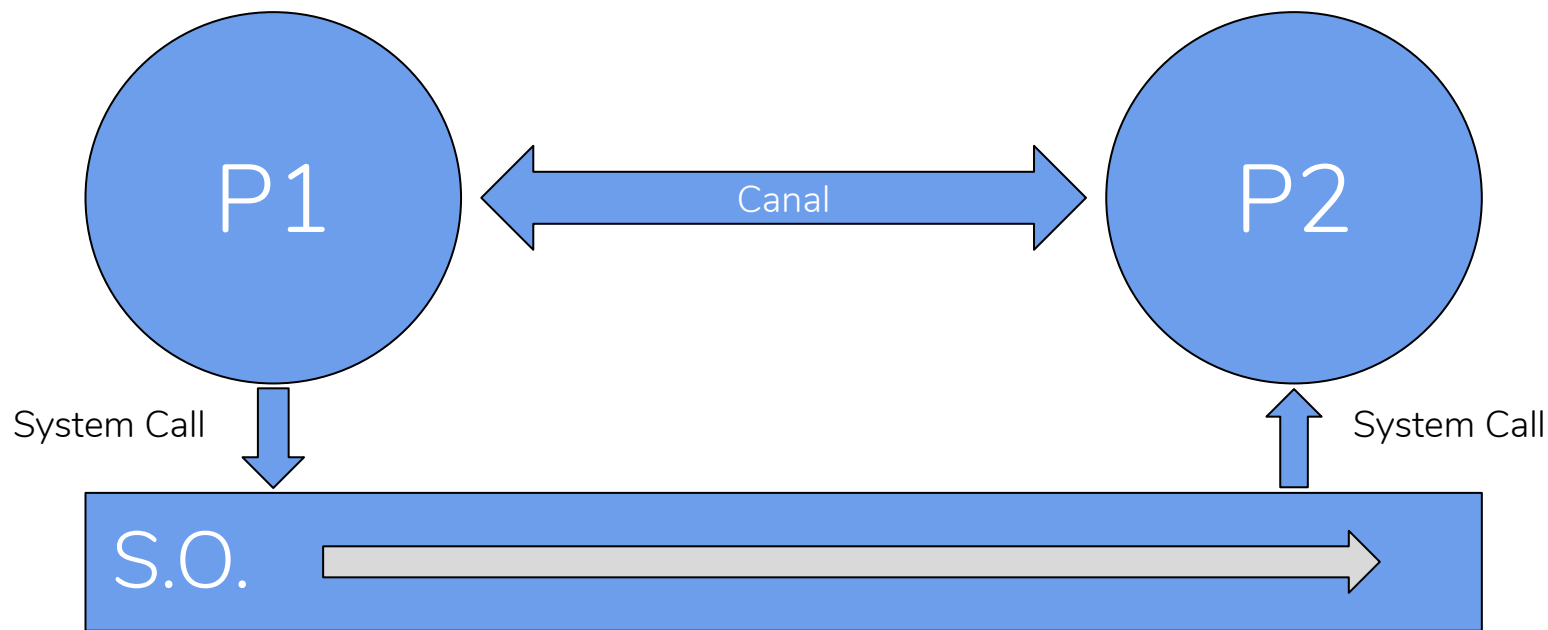
Atividade Prática

# Comunicação entre processos

- Processos executam em “cápsulas” autônomas;
- O *hardware* oferece proteção de memória, ou seja, um processo não acessa outro processo.



- Processos frequentemente necessitam comunicar-se/interagir;
- Essa comunicação é feita utilizando mecanismos de IPC: *Inter-Process Communication*;
- Essa comunicação é realizada através do S.O. através de canais de comunicação.



- Características desejáveis para comunicação:
  - Rapidez;
  - Simplicidade na utilização e implementação;
  - Modelo de sincronização bem definido;
  - Versatilidade;
  - Sem mudanças em ambientes distribuídos;
  - SINCRONIZAÇÃO.

- Sincronização é uma das maiores preocupações para comunicação entre os processos
  - Deve permitir que um processo que envia uma mensagem indique quando o dado foi transmitido;
  - Deve permitir que um processo que recebe uma mensagem saiba quando um dado está disponível;
  - Deve permitir que ambos saibam o momento exato em que podem realizar uma nova comunicação.

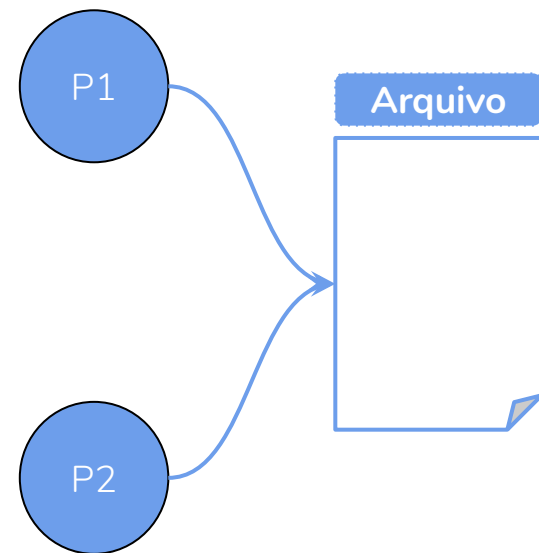
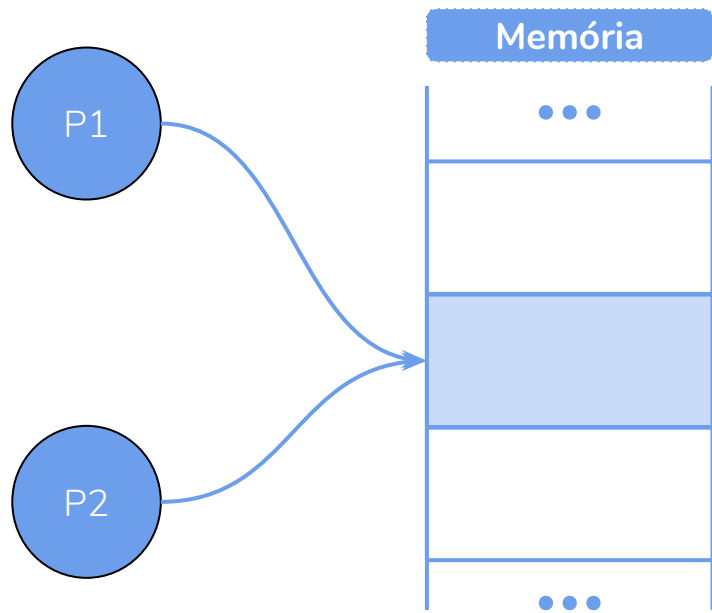


- Fundamentalmente duas abordagens:
  - Espaço de endereçamento cooperativo
    - Memória compartilhada.
  - Mecanismos do próprio Sistema Operacional para transportar dados de um processo a outro
    - Pipes;
    - Sinais;
    - Sockets;
    - Etc.

- Memória compartilhada
  - Mesmo segmento de memória encontra-se no espaço de endereçamento dos processos comunicando;
  - S.O. oferece chamadas permitindo a criação de uma área de memória compartilhada
    - Não se envolve diretamente com a sincronização e comunicação
  - Quando um processo realiza modificação, todos os outros que compartilham podem ver a alteração.

- Memória compartilhada
  - Vantagens
    - É possível acessar uma parte específica de uma estrutura de dados, e não necessariamente a estrutura completa (*Random Access*);
    - Forma mais rápida para dois processos trocarem dados.
  - Desvantagens
    - Não há mecanismo automático de sincronização.

- O local da memória compartilhada (RAM, arquivo) não muda a natureza da comunicação.

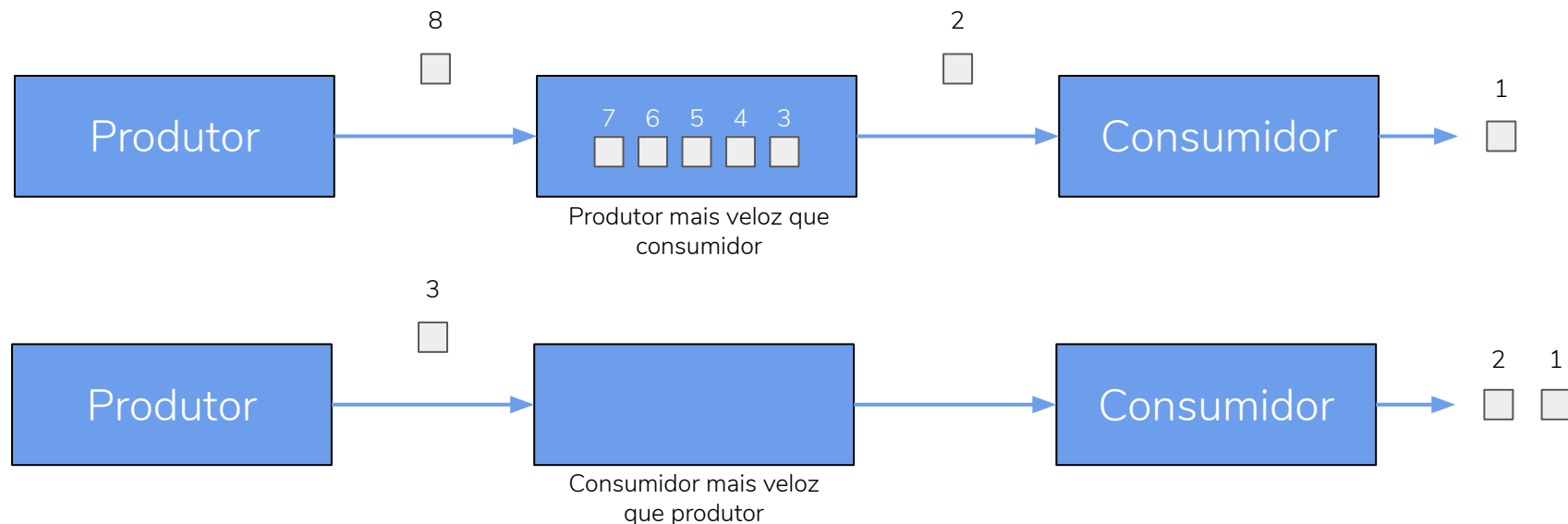


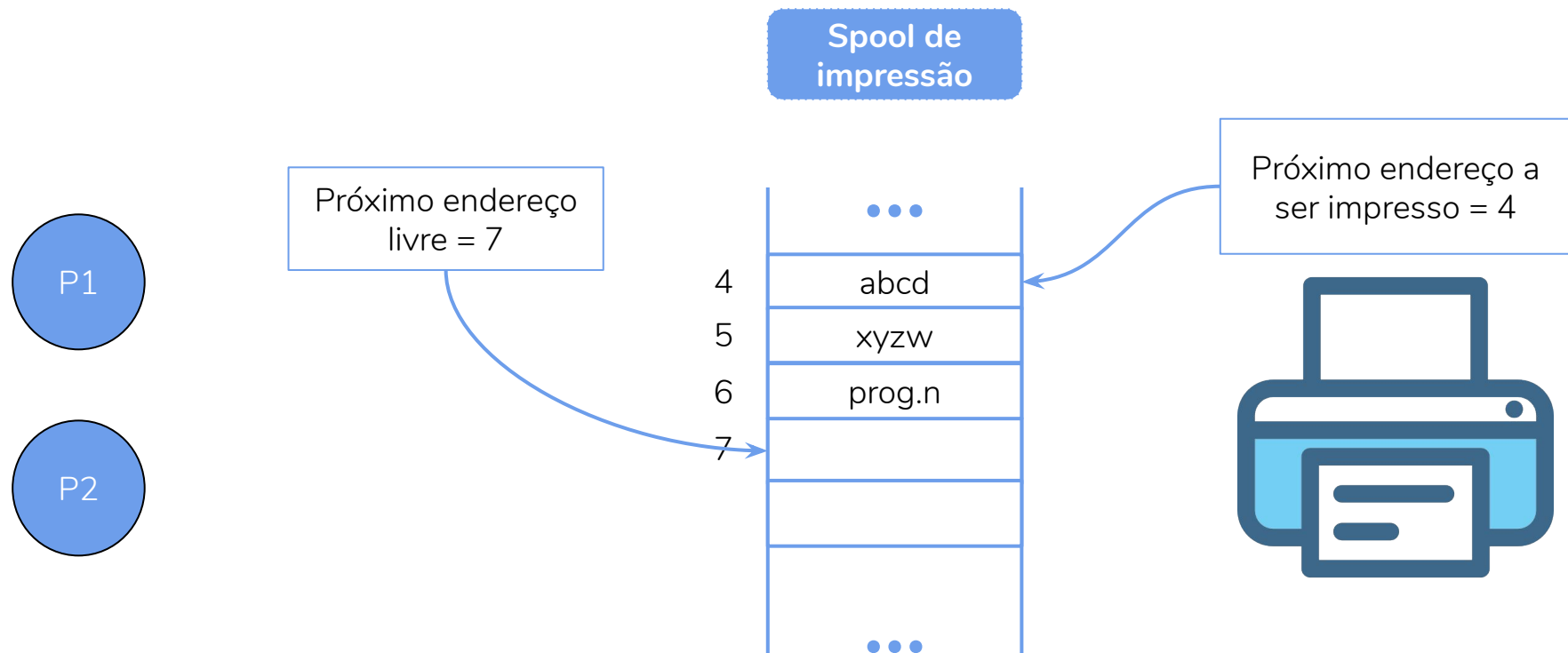
- Quais os problemas com essa técnica?



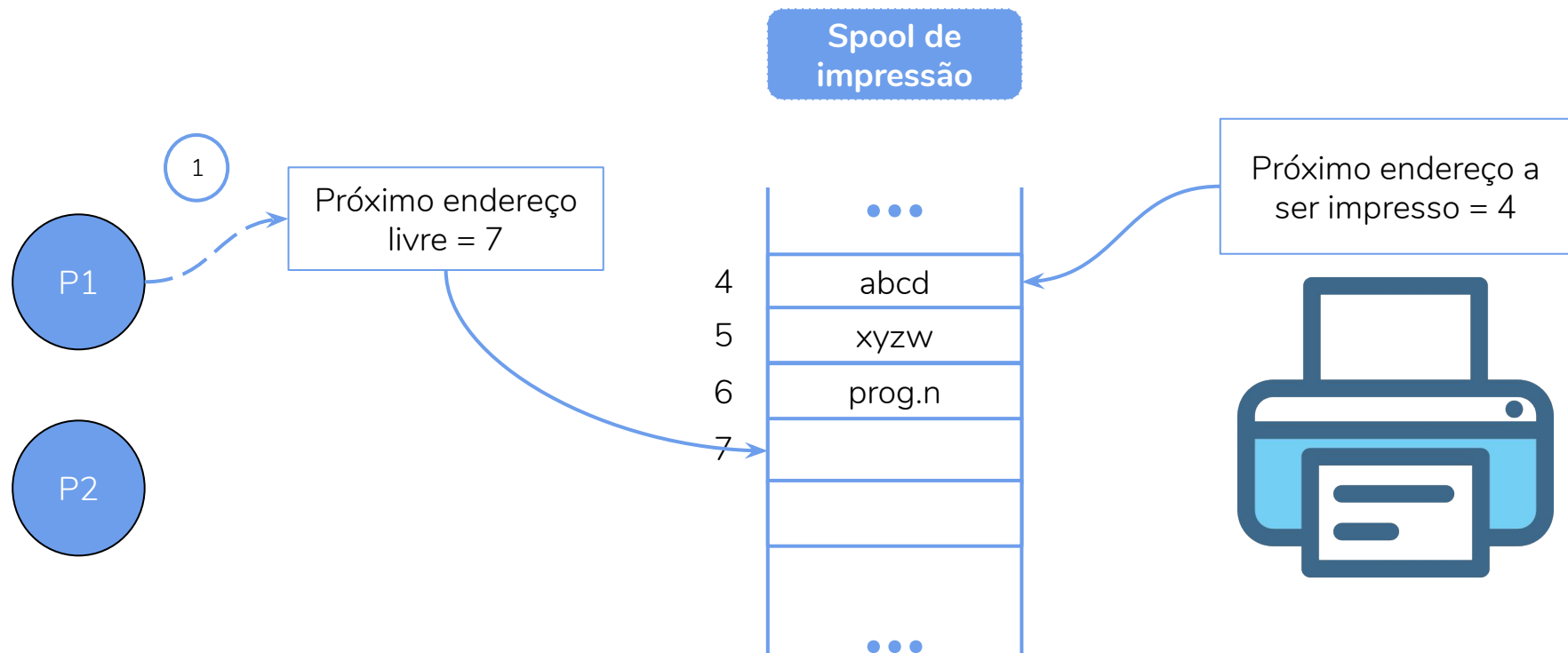
- Quais os problemas com essa técnica?
  - Condições de corrida;
    - Produtor mais rápido que o consumidor;
    - Consumidor mais rápido que o produtor.
  - Deadlock.

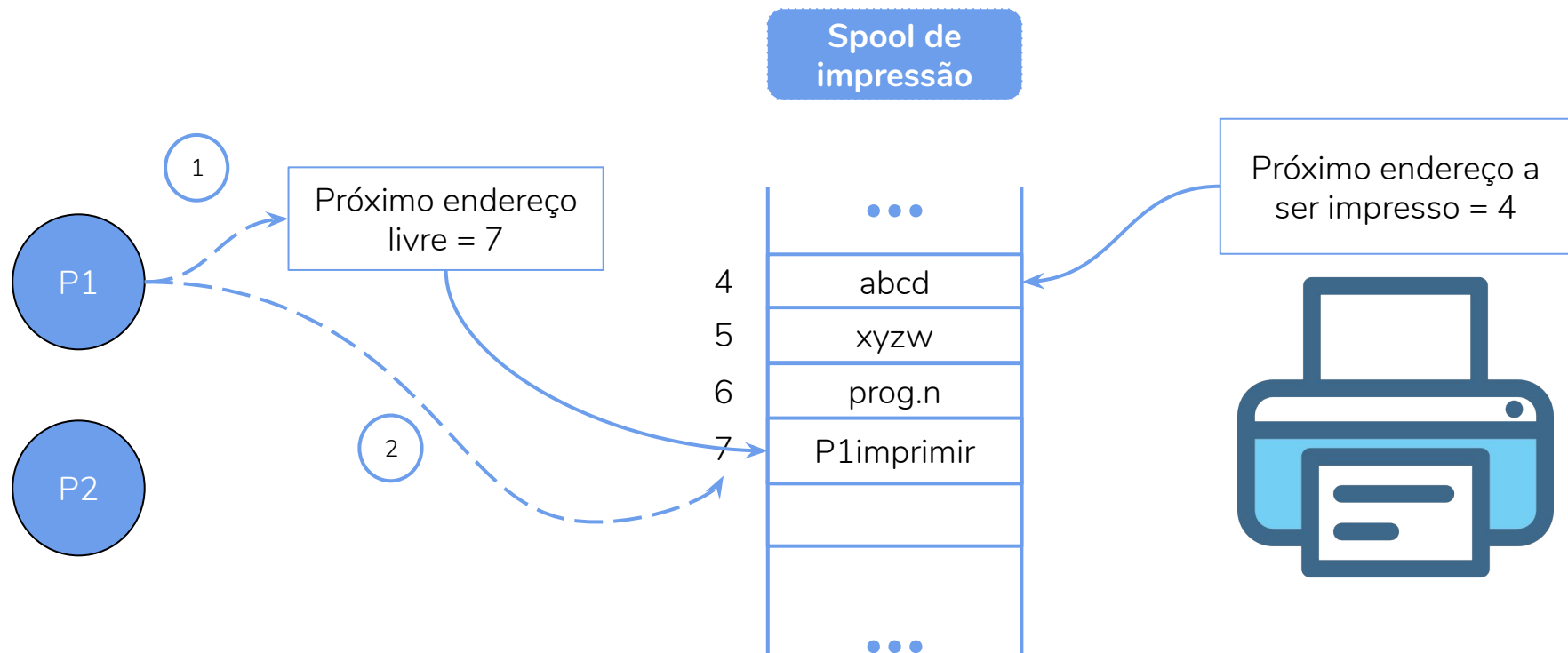
- Problema de sincronização
  - Produtor x Consumidor

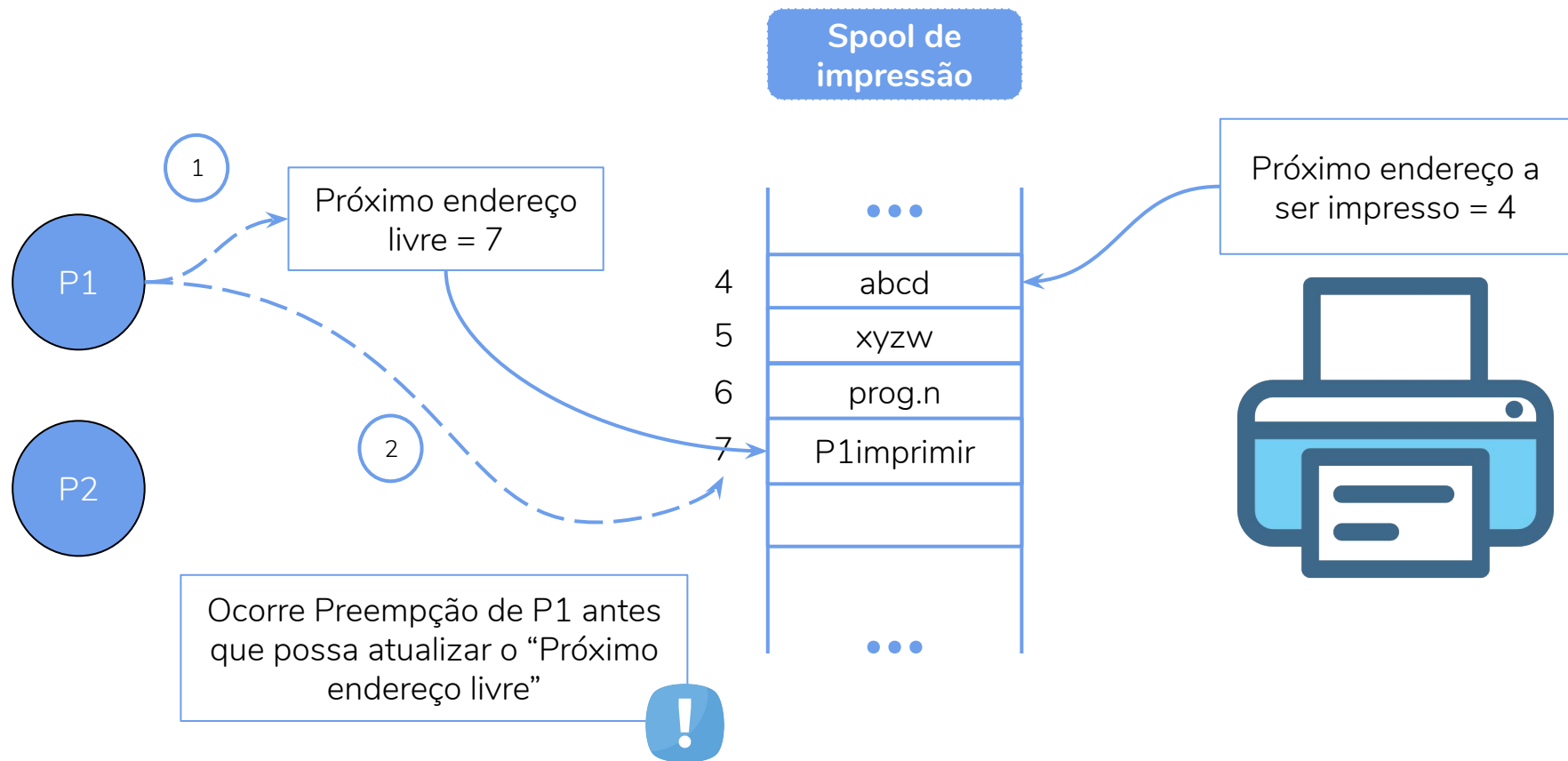


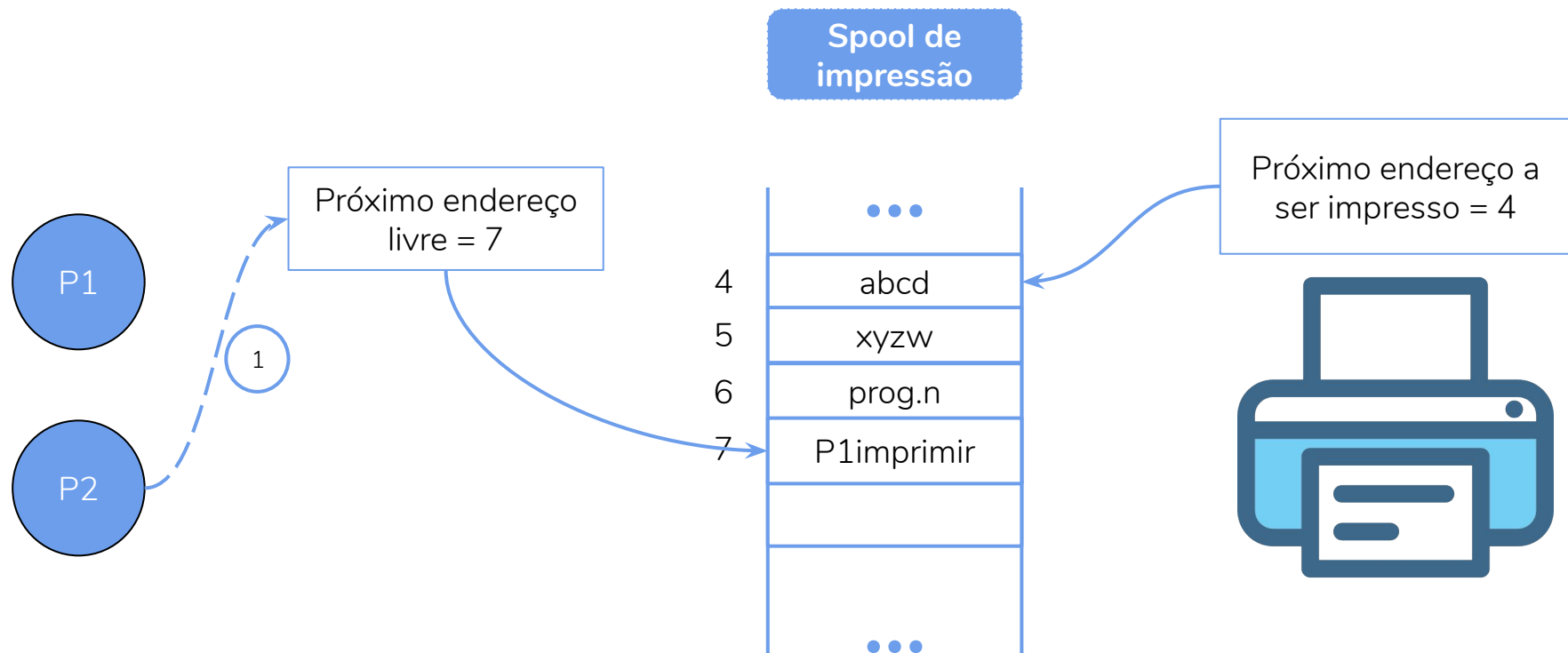


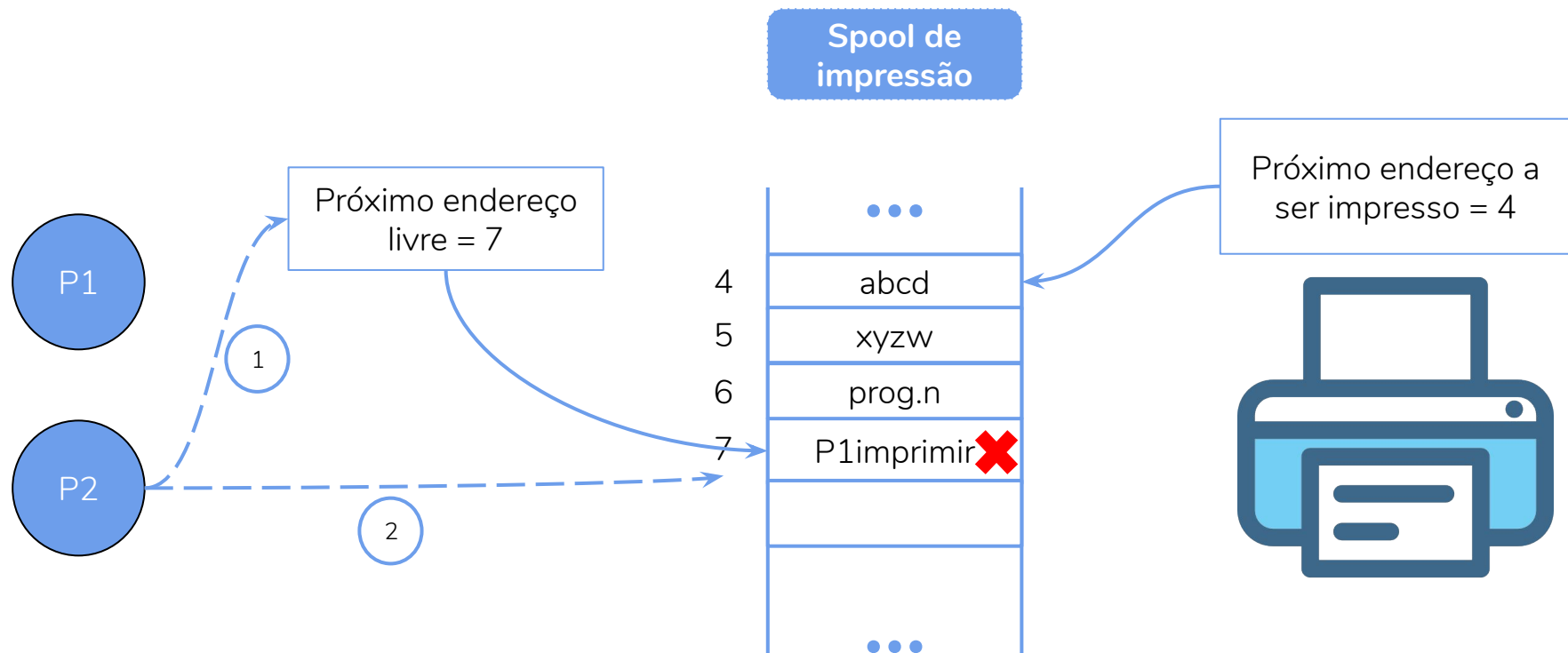


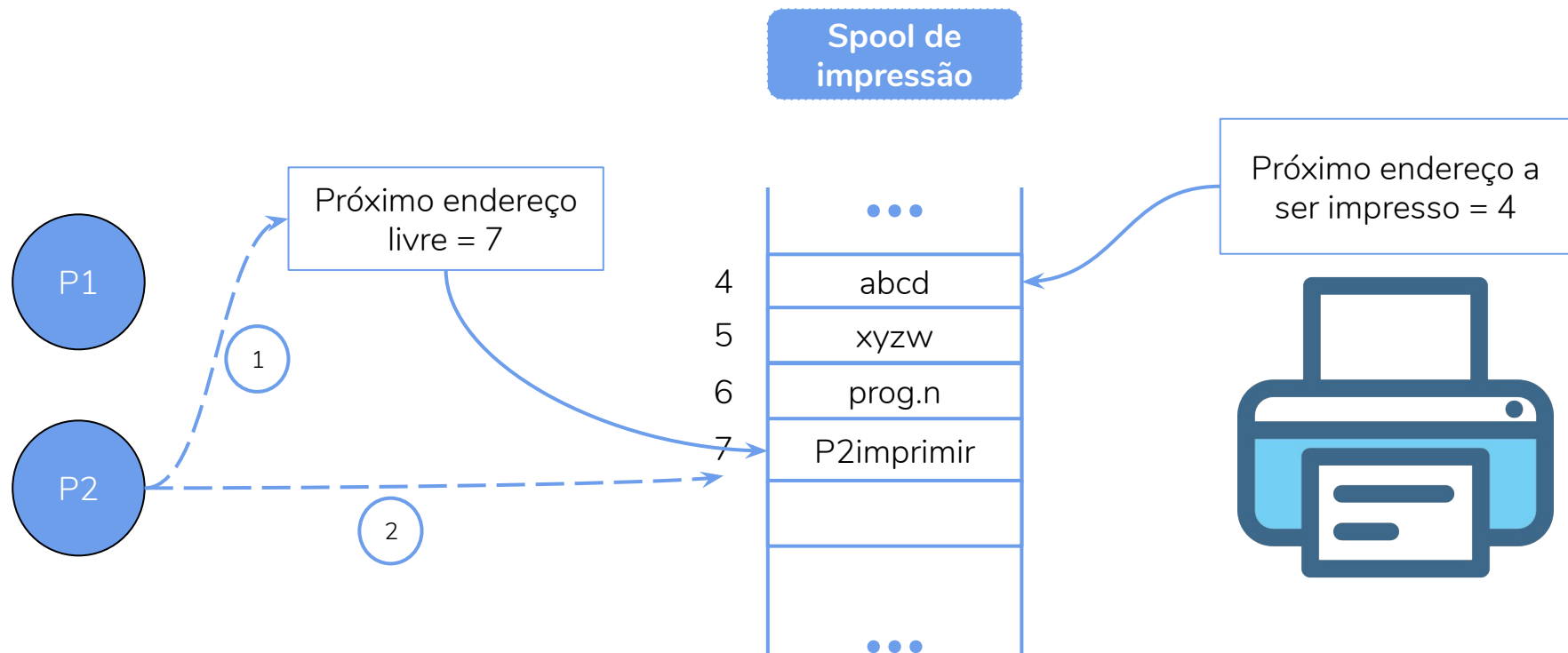


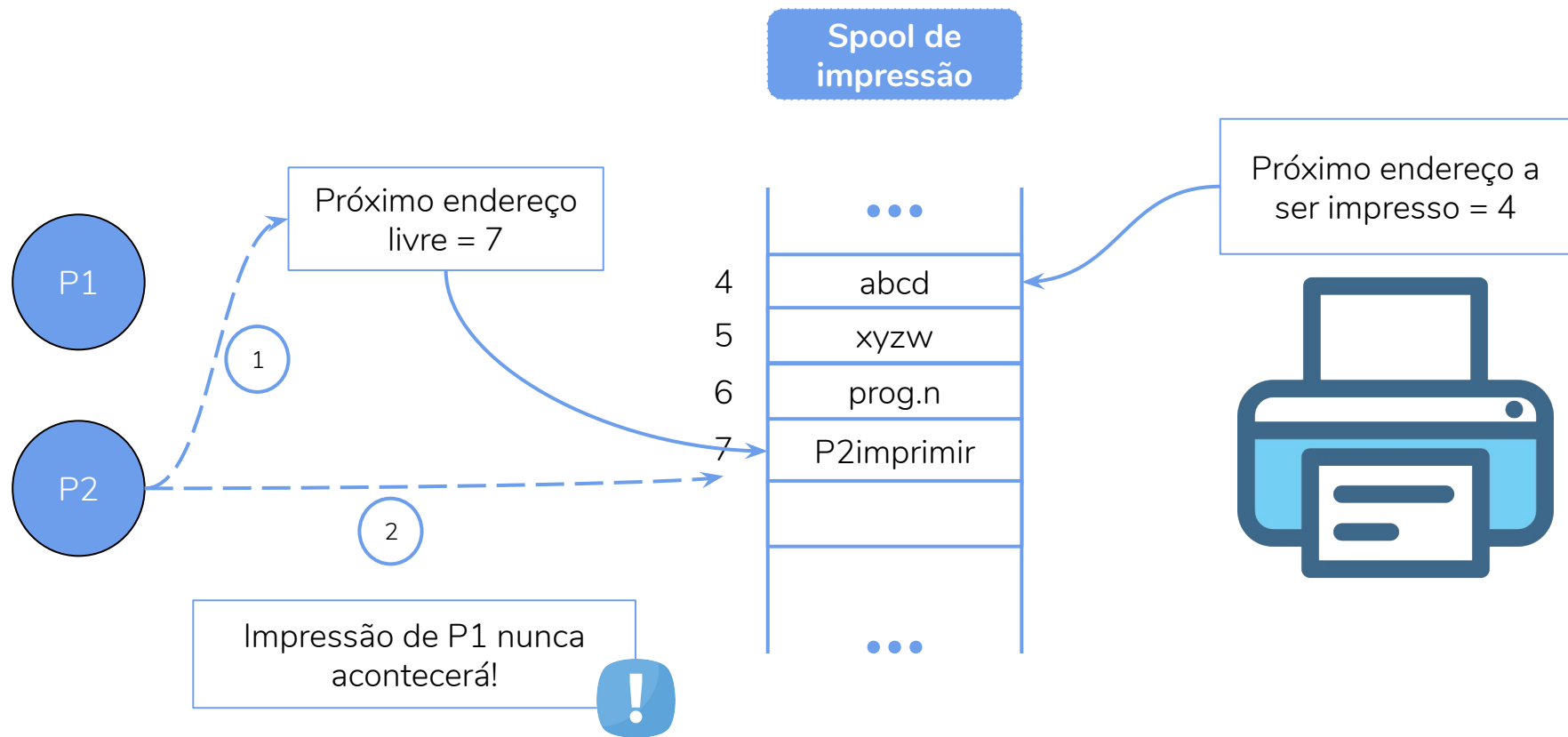












- Como resolver?

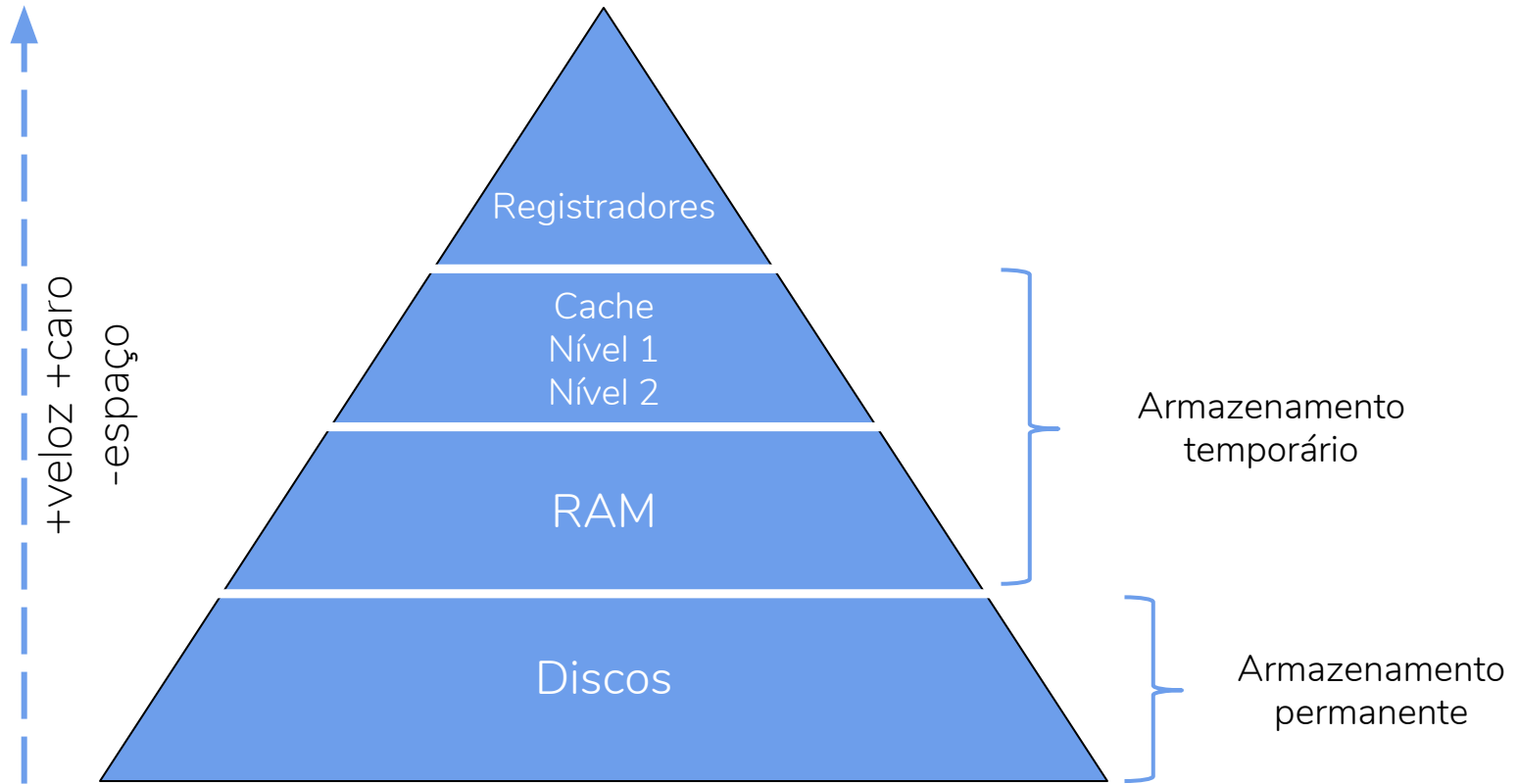


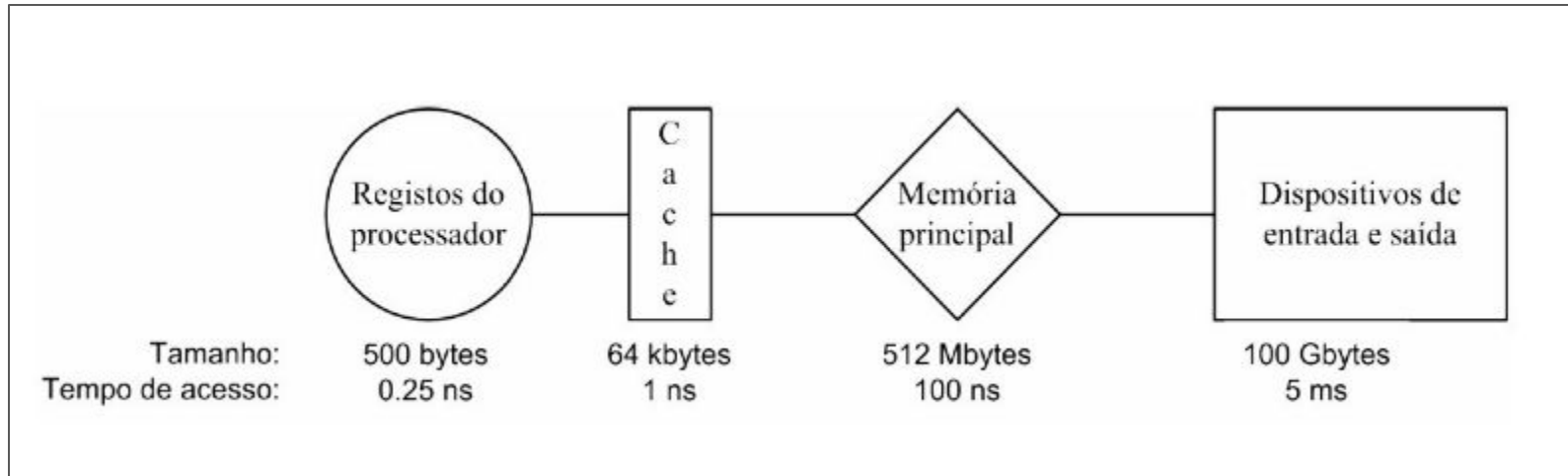


# Gerenciamento de Memória

- Historicamente a memória principal sempre foi vista como um recurso **escasso e caro**;
- Preocupação dos projetistas: desenvolver S.O. que não ocupassem muito espaço na memória e ao mesmo tempo otimizassem a utilização dos recursos;
- Mesmo com o barateamento do *hardware*, o gerenciamento de memória constitui um dos principais módulos do S.O.

- Principais funções do módulo de Gerenciamento de Memória:
  - Alocar memória para os processos;
  - Desalocar memória dos processos;
  - Gerenciar troca entre memória principal e secundária.





# Gerenciamento de Memória

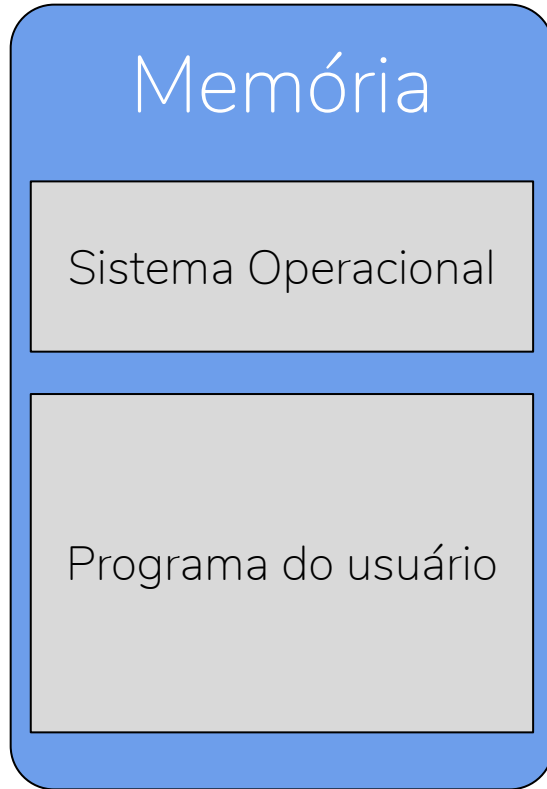
NÍVEL	1	2	3	4
NOME	Registos	<i>Cache</i>	Memória principal	Disco magnético
TAMANHO TÍPICO	< 1 kB	< 16 MB	< 16 GB	> 100 GB
TECNOLOGIA DE IMPLEMENTAÇÃO	CMOS, com portos múltiplos	SRAM CMOS No processador	DRAM CMOS	Disco magnético
TEMPO DE ACESSO (NS)	0.25 a 0.5	0.5 a 25	80 a 250	5 000 000
LARGURA DE BANDA (MB/S)	20 000 a 100 000	5000 a 10 000	1000 a 5000	20 a 150
GERIDO POR	Compilador	<i>Hardware</i>	Sistema operativo	Sistema operativo/ utilizador
APOIADA POR	<i>Cache</i>	Memória principal	Disco rígido	DVD, CD ou fita magnética

- Gerenciamento da memória é a ferramenta utilizada para permitir aos programas em execução utilizarem a memória do computador para armazenar
  - Instruções;
  - Dados que serão manipulados.

- Principais objetivos da gerência de memória
  - Disponibilizar área de armazenamento para processos executarem;
  - Proteger execução dos processos contra falhas;
  - Criar ambiente de execução com desempenho satisfatório;
  - Compartilhamento de memória entre processos;
  - Acesso transparente de memória por desenvolvedores.



- Gerenciamento básico de memória
  - Primeiros S.O.;
  - Monoprogramáveis;
  - ALOCAÇÃO CONTÍGUA;
  - Memória principal dividida entre o S.O. e o programa em execução.

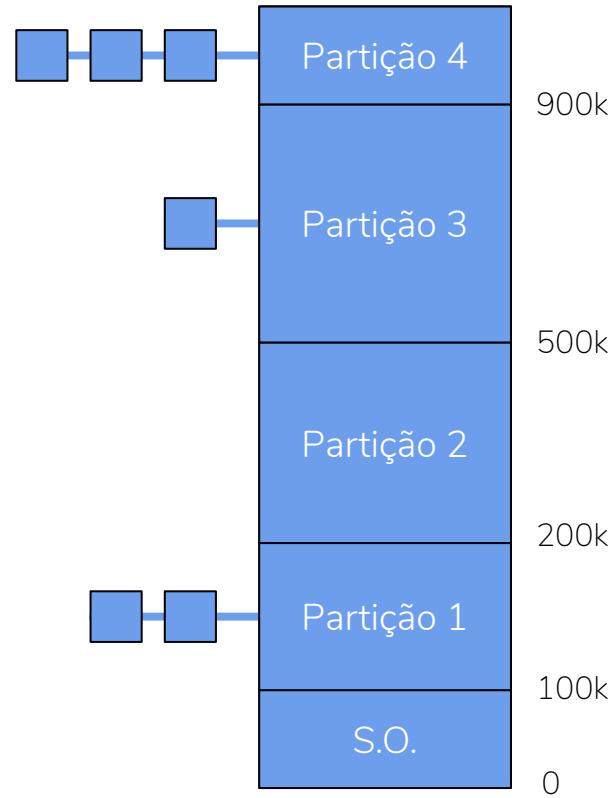


- Alocação contígua
  - Processo tem acesso a toda memória;
  - Sistemas monoprogramáveis.

- Maioria dos S.O. são multiprogramáveis (1+ processos);
- Forma mais simples:
  - Divisão da memória principal em partições estáticas;
  - Tamanhos definidos;
  - Tamanhos estabelecidos na inicialização do sistema operacional;
  - ALOCAÇÃO PARTICIONADA ESTÁTICA ou ALOCAÇÃO FIXA.

# Alocação Fixa

- Processo é colocado em uma fila na entrada de cada partição;
- Escolha da partição: menor partição capaz de armazená-lo;
- Caso o processo não ocupe o espaço total de sua partição, o restante é inutilizado.

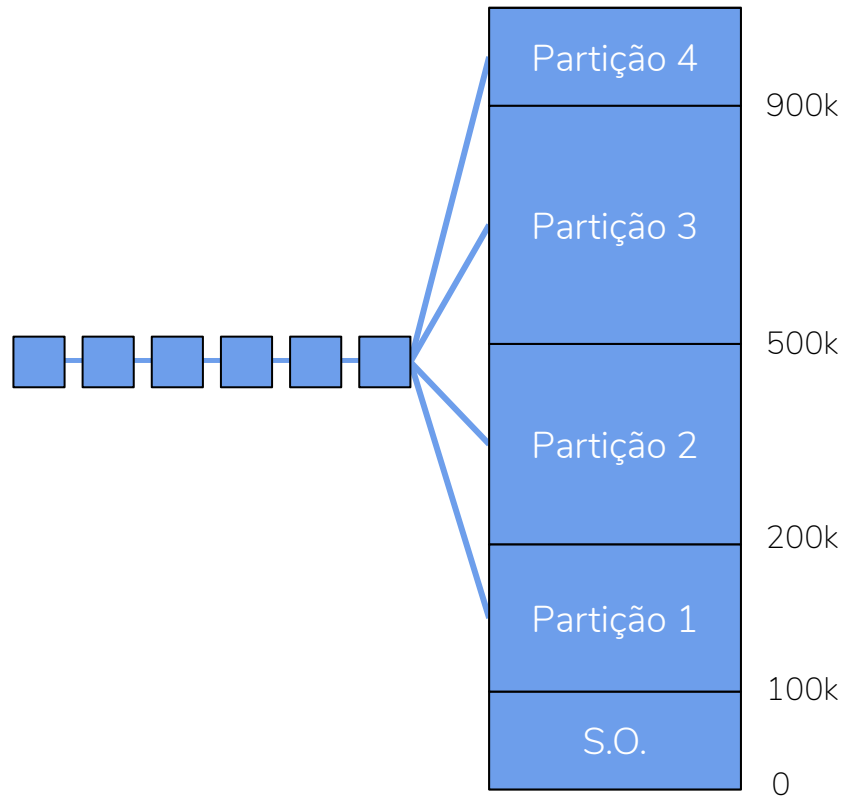


- Problemas:
  - Filas com grande quantidade de processos enquanto outras filas vazias ou com poucos processos;
  - Espaço desperdiçado nas partições;
  - Exemplo: Partição 4 com 3 processos e partição 2 vazia.
- Como resolver?



- Problemas:
  - Filas com grande quantidade de processos enquanto outras filas vazias ou com poucos processos;
  - Espaço desperdiçado nas partições;
  - Exemplo: Partição 4 com 3 processos e partição 2 vazia.
- Como resolver?
  - Implementar o método com uma fila única.





- Quando determinada partição está livre, o mais próximo do início da fila que melhor se ajusta à essa partição é carregado.

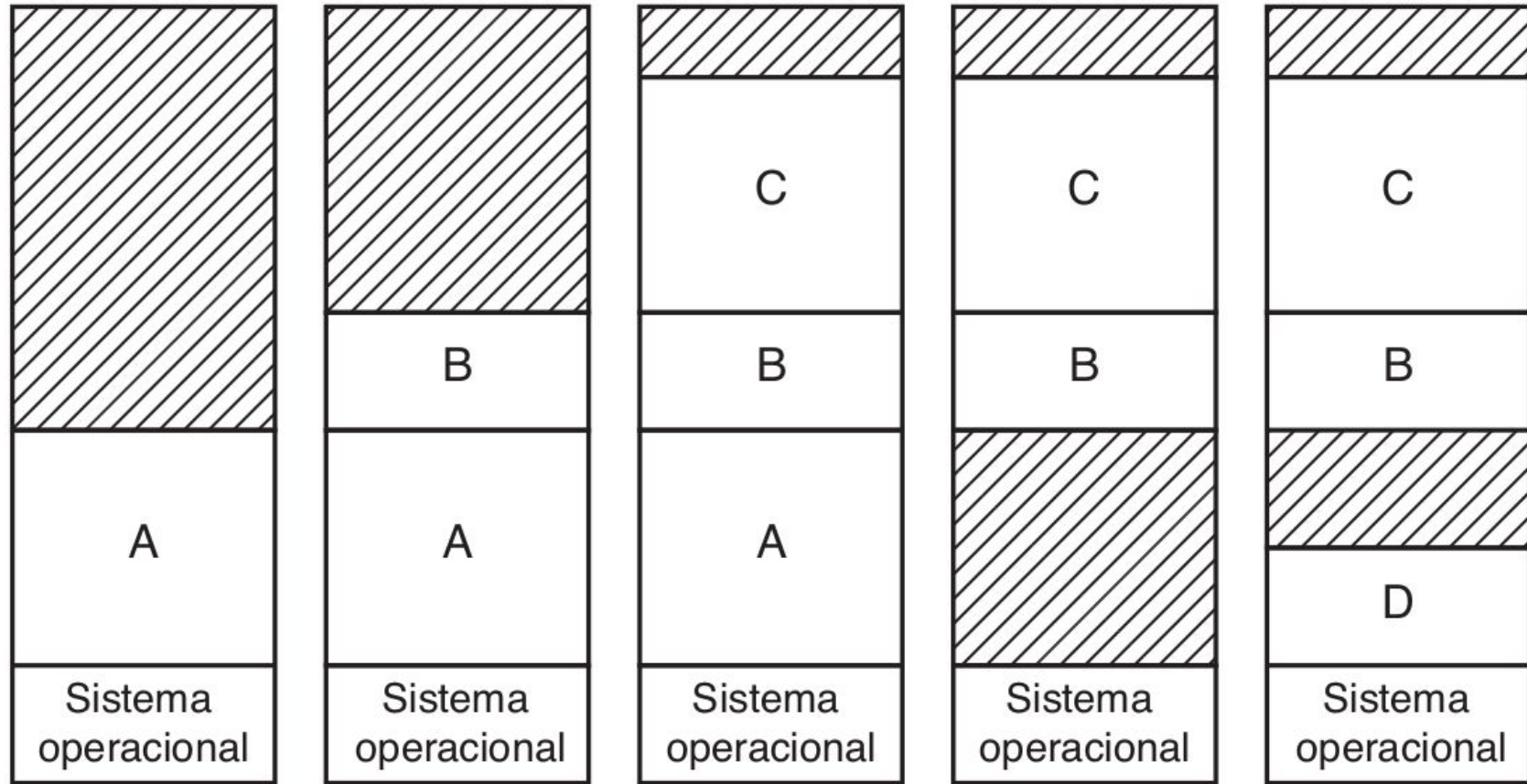
- Problemas
  - Grande desperdício de memória;
  - Processo pode ocupar somente um pequeno pedaço de uma partição;
  - Restante da partição fica inutilizado.
- Como resolver?



- Problemas
  - Grande desperdício de memória;
  - Processo pode ocupar somente um pequeno pedaço de uma partição;
  - Restante da partição fica inutilizado.
- Como resolver?
  - Alocação com partições variáveis

# Alocação Variável

- Tamanho das partições ajustado dinamicamente quando os processos chegam na memória;
- O processo utilizará um espaço de memória necessário, tornando sua partição;
- Não há mais problemas de fragmentação interna na partição, pois serão definidas a partir da quantidade total alocada pelo processo.



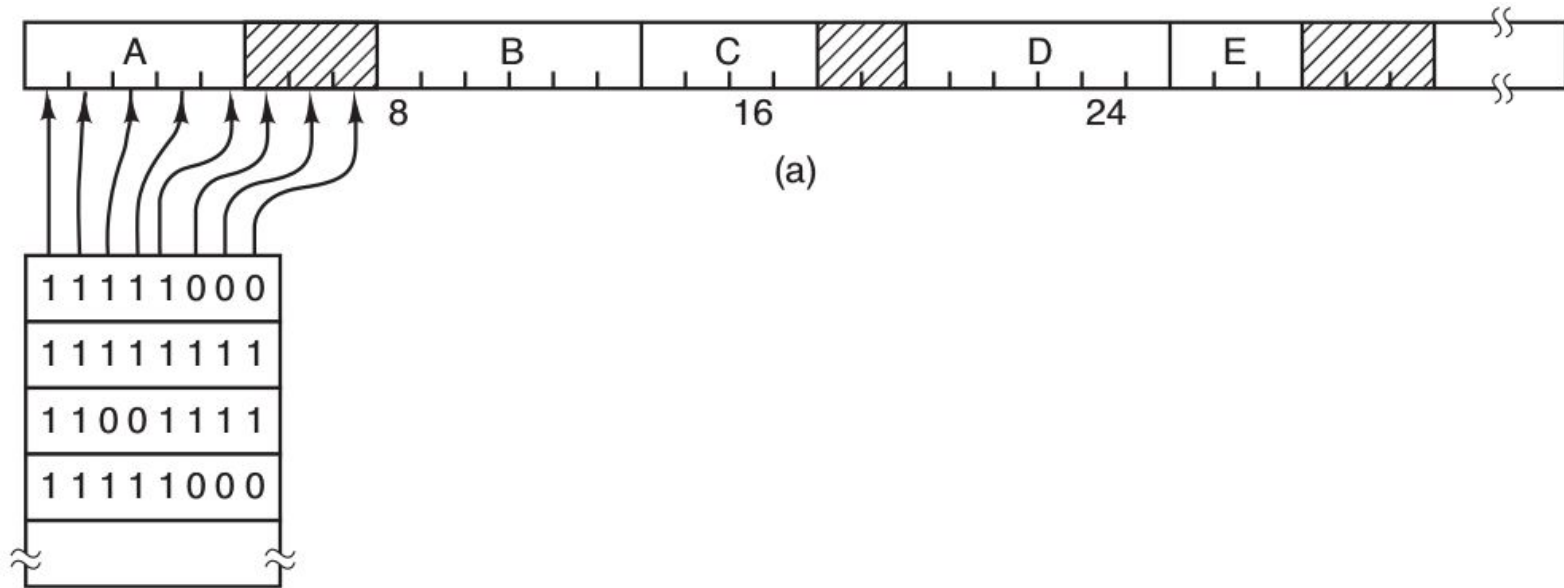
- Processos definem o tamanho de sua partição ao chegarem na memória;
- Quando não há mais espaço disponível, o processo mais “antigo” na memória é retirado para dar espaço para o próximo.
- O processo de retirar um processo de memória, atualizar em disco e colocar outro processo no lugar é chamado de TROCA.

- A TROCA consiste em trazer um processo inteiro, executá-lo temporariamente e então devolvê-lo ao disco;
- Isso acontece quando a memória principal disponível é insuficiente para manter todos os processos carregados
  - Todos os processos na memória principal seria a solução ótima.
- Outra estratégia que permite que apenas parte do programa fique em memória principal: Memória virtual.



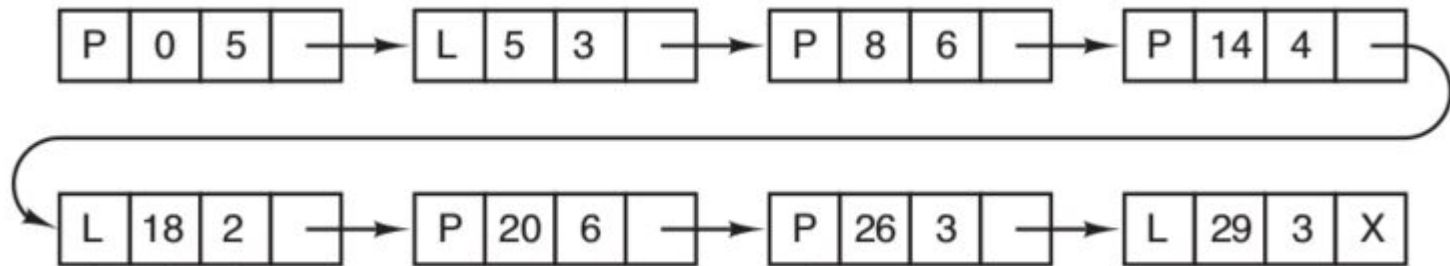
- Mais flexível que o método com partição estática;
- Flexibilidade complica a tarefa de alocar e desalocar a memória;
- Complica o monitoramento da memória utilizada
  - Os espaços alocados não são mais fixos, por isso não há como saber antecipadamente a quantidade de memória utilizada.
- Há 2 formas de monitorar a memória utilizada
  - Mapa de bits;
  - Lista encadeada.

- Memória é dividida em unidades de alocação;
- Para cada unidade é associado um bit no mapa;
- Valor 0: livre, valor 1: ocupado.



- Ponto crucial: definição do tamanho da unidade de alocação.
  - Unidade muito pequena: mapa de bits muito grande;
  - Unidade muito grande: mapa de bits menor, porém uma quantia de memória poderá ser desperdiçada na última unidade se o tamanho do processo não for um múltiplo exato da unidade de alocação.

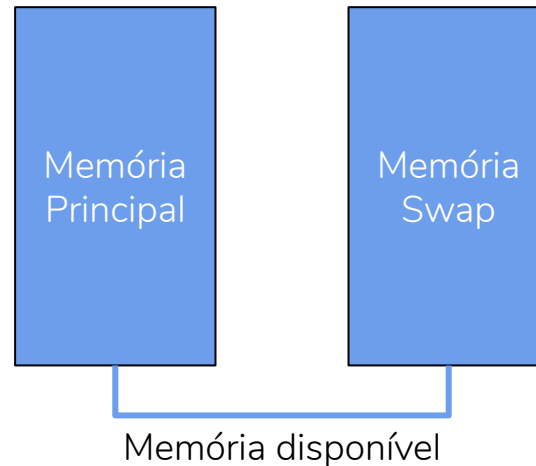
- Lista encadeada dos segmentos alocados e livres;
- Pode ser um processo ou uma lacuna entre dois processos;
- Cada nó da lista é formado por uma entrada especificando se é um P (processo) ou L (lacuna), o endereço onde se inicia a quantidade de unidades de alocação e um ponteiro para a próxima entrada.



# Memória Virtual

- Bloatware: Softwares que usam quantidades excessivas de memória.
- Anos 1980, muitas universidades executavam um sistema de compartilhamento de tempo com dúzias de usuários executando simultaneamente em um VAX de 4 MB;
- Agora a Microsoft recomenda no mínimo 2 GB para o Windows 10 de 64 bits. A tendência à multimídia coloca ainda mais demandas sobre a memória;
- Lei de Parkinson.

- Estender a memória principal através da memória secundária;
- Dá a impressão ao usuário de uma maior quantidade de memória principal disponível;
- Também chamado de memória de swap.



**/dev/sda - GParted**

GParted Editar Ver Dispositivo Partição Ajuda













Novo Excluir Redimensionar/Mover Copiar Colar Desfazer

/dev/sda (93.16 GB)

/dev/sda1  
35.12 GB

/dev/sda7  
29.12 GB

/dev/sda2  
19.99 GB

Partição	Sistema de arquivos	Ponto de Montagem	Rótulo	Tamanho	Utilizado	Não utilizado	Flags
/dev/sda1	 	ntfs		35.12 GB	---	---	boot
▼ /dev/sda3	 	extended		38.04 GB	---	---	
/dev/sda5	 	/	Set	7.92 GB	4.48 GB	3.44 GB	
/dev/sda6	 	linux-swap		1.00 GB	---	---	
/dev/sda7	 	/home	Set	29.12 GB	23.60 GB	5.52 GB	
/dev/sda2		linux-swap		19.99 GB	---	---	
não alocado		não alocado		7.84 MB	---	---	

0 operações pendentes



img memória virtual windows

# Paginação

- Esquema de gerenciamento de memória em que o espaço de endereçamento físico de um processo não é contíguo;
- A paginação evita a fragmentação gerada pela alocação dinâmica;
- Programas são capazes de gerar endereços virtuais;
- Conjunto de endereços virtuais formam o espaço de endereçamento virtual.

- Os endereços virtuais não vão diretamente para o barramento da memória;
- Passam antes por uma unidade chamada MMU (Memory Management Unit – Unidade de gerenciamento de memória);
- A MMU mapeia um endereço virtual para um endereço lógico;
- O endereçamento virtual é dividido em unidades chamadas de páginas.

# Gerenciamento de Arquivos

DEITEL, Harvey M.; DEITEL, Paul J.; CHOFFNES, David R. Sistemas operacionais. 3. ed. São Paulo: Pearson, 2005

SILBERSCHATZ, Abraham; GALVIN, Peter Baer; GAGNE, Greg. Fundamentos de sistemas operacionais – princípios básicos. 9. ed. Rio de Janeiro: LTC, 2013

TANENBAUM, Andrew S. Sistemas operacionais modernos. 3. ed. São Paulo: Pearson, 2008

MAZIERO, Carlos Alberto; Sistemas Operacionais: conceitos básicos, 2011

ZAGARI, Eduardo Nicola; Escalonamento de Processos

# Referências

Icons made by Freepik from [www.flaticon.com](http://www.flaticon.com)

Icons made by Turkkub from [www.flaticon.com](http://www.flaticon.com)

Icons made by Icon Pond from [www.flaticon.com](http://www.flaticon.com)

Icons made by Pixel Perfect from [www.flaticon.com](http://www.flaticon.com)

Icons made by Vectors Market from [www.flaticon.com](http://www.flaticon.com)

Icons made by SmashIcons from [www.flaticon.com](http://www.flaticon.com)

Icons made by RoundIcon from [www.flaticon.com](http://www.flaticon.com)

Icons made by Drycons from [dryicons.com](http://dryicons.com)