



SÃO
PAULO
TECH
SCHOOL

Sistemas Operacionais

Gerenciamento de processos

Marcio Santana

marcio.santana@sptech.school

- **Todo processo** é um **programa em execução**, e cada processo tem um conjunto de **estados**
- O **SO é responsável** pelo gerenciamento de todas as informações necessárias para a **execução de qualquer programa**
- **Processos** são entidades **independentes**. Cada um possui permissões de acesso e atributos.
- No Linux, um **processo é uma thread de um software** em execução.
- Em outros sistemas, os **processos** também ganham o nome de **tarefas** (tasks)
- No Windows, temos o Gerenciador de Tarefas, que auxilia por exemplo a fechar um aplicativo que travou ou está em estado indesejável. Igualmente podemos gerenciar os processos no Linux

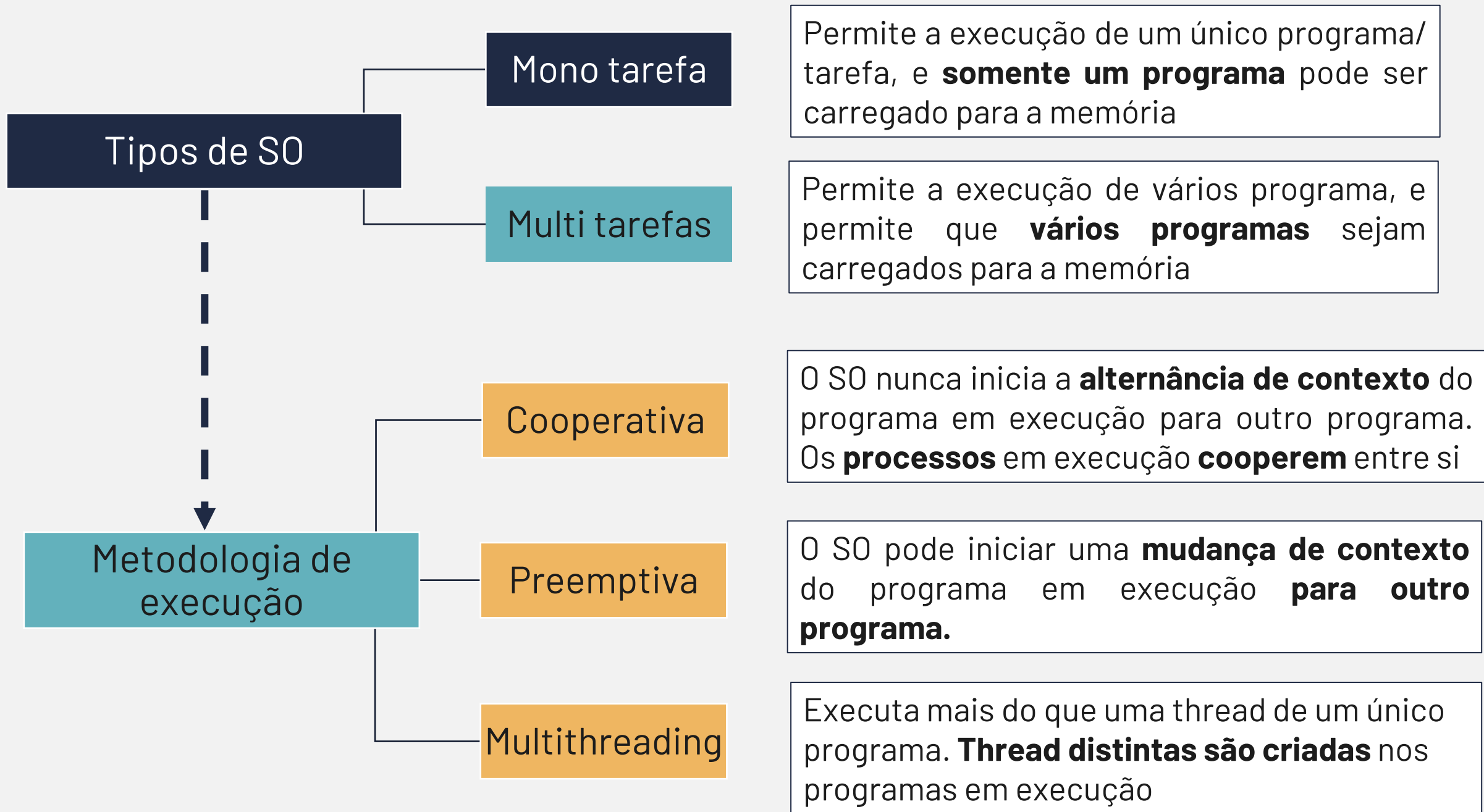
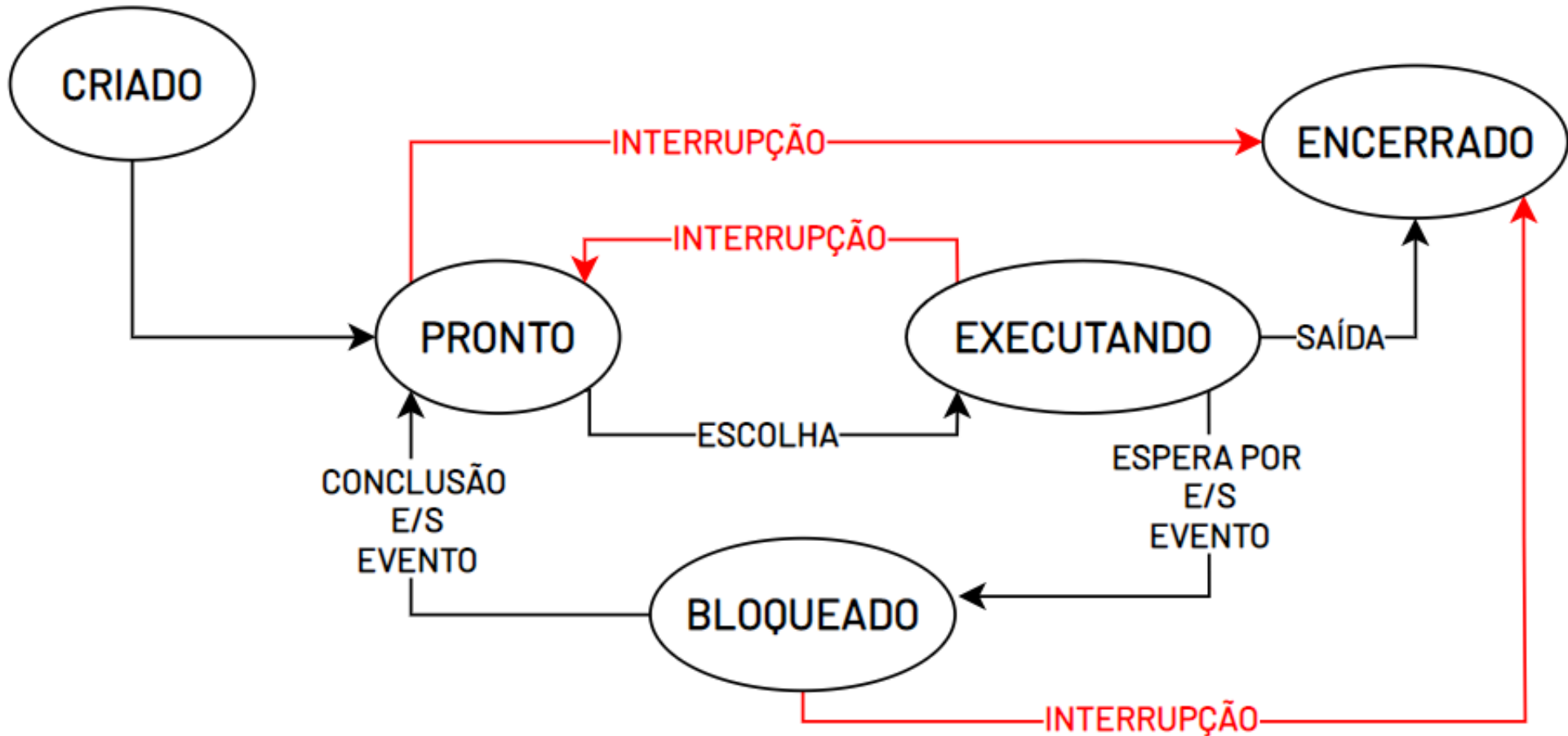


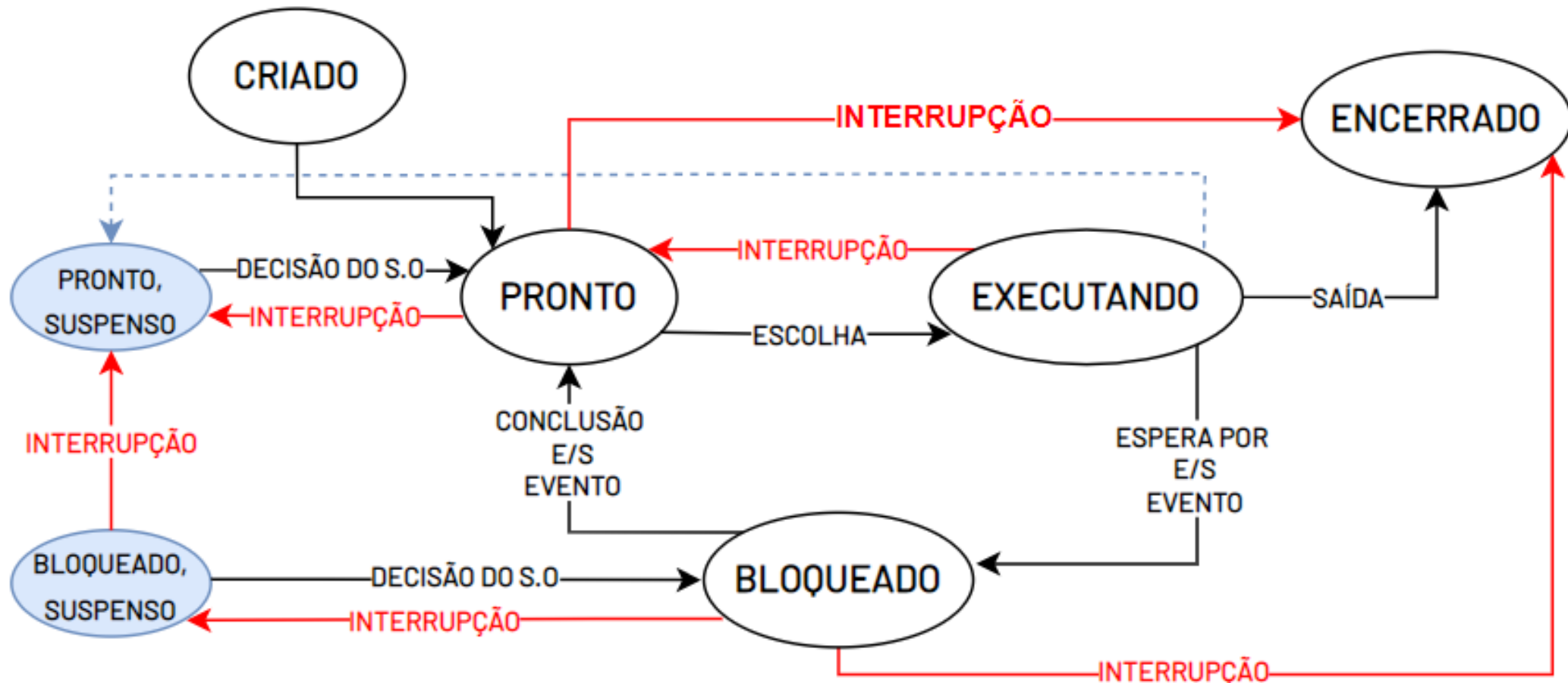
Diagrama de estados – ciclo de vida do Processo



(Silberschatz, 4.1.2)

Diagrama de estados – ciclo de vida do Processo

Transições com estado Suspenso



- **O kernel alterna entre threads em um esforço para compartilhar a CPU efetivamente, essa atividade é chamada de alternância de contexto**
- Quando uma thread é executada pela duração de seu intervalo de tempo ou quando é bloqueado porque requer um recurso indisponível no momento, o **kernel encontra outra thread para executar**.
- O SO também pode interromper o thread em execução no momento para executar uma thread acionado por um evento.
- Além disso, as alternâncias de contexto entre processos são classificadas como **voluntárias** ou **involuntárias**.
 - Uma opção de contexto voluntária ocorre quando um encadeamento é bloqueado porque requer um recurso indisponível
 - Uma alternância involuntária de contexto ocorre quando um encadeamento é executado pelo período de tempo ou quando o sistema identifica um encadeamento de prioridade mais alta a ser executado

Escalonamento de processos

Quando um computador é multiprogramado, ele muitas vezes tem **múltiplos processos** que **competem** pela **CPU** ao mesmo tempo.

Essa situação ocorre quando dois ou mais processos estão simultaneamente no estado **pronto**.

Se somente uma CPU se encontrar disponível, deverá ser feita uma escolha de qual processo executará em seguida.

A parte do SO que faz a escolha é chamada de **escalonador**, e o algoritmo que ele usa é o **algoritmo de escalonamento**.

Objetivos do algoritmo de escalonamento

Responder rapidamente às requisições

Evitar a perda de dados

Minimizar o tempo entre a submissão e o término do processo

Manter a CPU ocupada o tempo todo

Verificar se a política estabelecida é cumprida

Maximizar o número de tarefas por hora

Dar a cada processo uma porção justa de CPU

Manter ocupadas todas as partes do sistema

Tipos de escalonamentos

- ❑ **FCFS:** *first come, first served (FIFO)*. Com esse algoritmo, a CPU é atribuída aos processos na ordem em que eles a requisitam. Basicamente, há **uma fila única de processos prontos**
- ❑ **SJF:** *Shortest Job First* ou tarefa mais curta primeiro. Se supõem como previamente conhecidos todos os tempos de execução, muito utilizado em lote. Quando várias tarefas igualmente importantes estiverem postadas na fila de entrada à espera de serem iniciados o **escalonador escolhe a tarefa mais curta primeiro**
- ❑ **Round-Robin:** Escalonamento por chaveamento circular. Divide o tempo da CPU em fatias iguais, chamadas de **quantum**, e atribui uma fatia para cada processo na fila de prontos, de forma circular. Se o processo não terminar dentro do seu quantum, ele é interrompido e volta para o final da fila, dando a vez ao próximo processo

Tipos de escalonamentos

- ❑ **Prioridade preemptivo:** permite que o **escalonador interrompa a execução de um processo e substitua-o por outro**, caso ocorra algum evento de maior prioridade, como a chegada de um novo processo, ou o término de uma operação de entrada/saída
- ❑ **Por prioridade:** seleciona o processo que possui a **maior prioridade na fila de prontos**, sendo que a prioridade pode ser estática ou dinâmica, dependendo de fatores como o tempo de espera, o tipo de processo ou a urgência do usuário
- ❑ **Loteria:** atribui um número de bilhetes para cada processo na fila de prontos, de acordo com **algum critério**, como a prioridade, o tamanho ou o tempo de execução. Em seguida, sorteia um bilhete aleatoriamente e seleciona o processo que o possui, concedendo-lhe a CPU

Composição de um processo

Os **processos** apresentam um conjunto de característica e atributos

- ❑ Proprietário / grupo do processo
 - Compartilhamento de recursos
 - Um processo pai cria processos filhos
 - Os filhos podem executar o mesmo código, ou trocá-lo
 - Obtêm-se uma árvore de processos
 - Implica na definição semântica de término de um processo – toda sua descendência morre
- ❑ Estado do processo (em espera, em execução, etc)
- ❑ Prioridade de execução
- ❑ Recursos de memória

Comandos de gerenciamento de processos no Linux

Execute no terminal

\$ **top**

O comando top é a maneira mais comum de verificar o uso de processos do sistema e quais deles estão consumindo mais memória ou processamento.

1.

Ready (scheduler tasks kernel)
2.

Running
3.

Sleeping (not use)
4.

Stopped
5.

Zombie (PPID is kill)
6.

Waiting (thread)
7.

Dead (comand kill)

```
ubuntu@eduardo-server: ~  
top - 18:39:15 up 2 min, 2 users, load average: 0.72, 0.52, 0.21  
Tasks: 108 total, 1 running, 107 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni,100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st  
MiB Mem : 1964.0 total, 863.9 free, 582.2 used, 517.8 buff/cache  
MiB Swap: 850.0 total, 850.0 free, 0.0 used. 1229.4 avail Mem  
  
  PID USER  PR  NI  VIRT  RES  SHR S  %CPU  %MEM    TIME+  COMMAND  
 1329 ubuntu 20   0  10520 4084 3480 R   6.2   0.2   0:00.01 top  
    1 root    20   0 100828 11632 8300 S   0.0   0.6   0:02.92 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
```

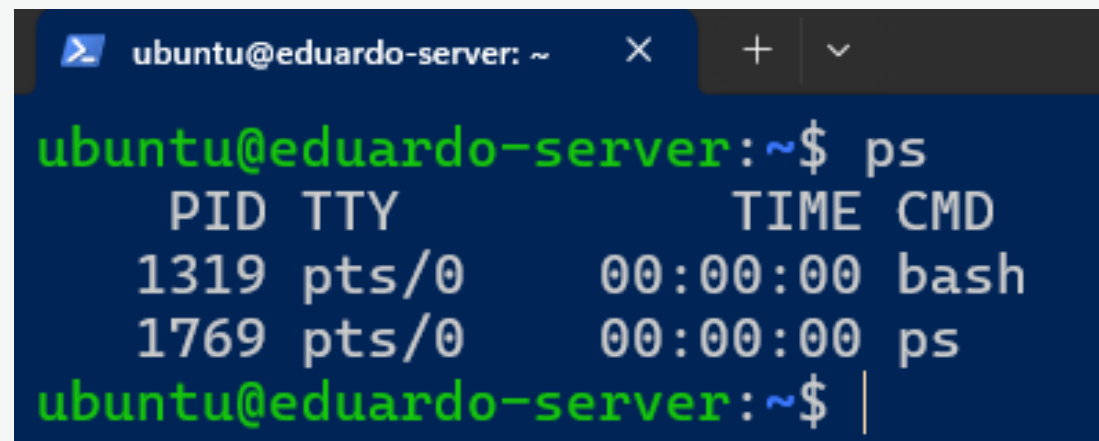
Vamos agora recordar sobre o comando de informações dos processos

\$ **ps**

Exibe informações sobre os processos ativos

ps [opções]

- **-a** exibe informações de outros usuários
- **-u** exibe o nome do usuário e a hora de início
- **-x** exibe processos não associados ao terminal
- **-l** exibe linhas detalhadas
- **-e** exibe todos os processos ativos

A terminal window titled 'ubuntu@eduardo-server: ~' with standard window controls. The prompt is 'ubuntu@eduardo-server:~\$'. The command 'ps' has been entered, and the output is displayed in a table format with columns: PID, TTY, TIME, and CMD. The output shows two processes: PID 1319, TTY pts/0, TIME 00:00:00, CMD bash; and PID 1769, TTY pts/0, TIME 00:00:00, CMD ps. The prompt is now 'ubuntu@eduardo-server:~\$ |' with a cursor.

```
ubuntu@eduardo-server:~$ ps
  PID TTY          TIME CMD
 1319 pts/0    00:00:00 bash
 1769 pts/0    00:00:00 ps
ubuntu@eduardo-server:~$ |
```

\$ **ps -aux**

- ❑ D - Uninterruptible sleep (usually IO) – processo em modo sleeping ininterrupto (em geral relativos a E/S)
- ❑ R - Running – rodando ou em execução (na fila de execução).
- ❑ S - Interruptible sleep (waiting for an event to complete) – Interrupção momentânea (em
- ❑ geral enquanto aguarda a conclusão de um evento.
- ❑ T - Stopped, interrompido por um sinal de controle ou por causa de algo que é rastreado.
- ❑ X - dead (should never be seen) - morto
- ❑ Z - Defunct ("zombie") Processo morto, relativo ao processo pai.

\$ **ps -aux (cont...)**

Os subcaracteres são:

- ❑ N -low-priority (nice to other users) – baixa prioridade, fornecendo-a processos de outros usuários
- ❑ L - has pages locked into memory (for real-time and custom IO) – mostra que há páginas bloqueadas na memória.
- ❑ s - is a session leader – mostra se a sessão é líder. Exemplo do shell ou de um bash. Ambos são pai porque podem ser executados por diferentes grupos de usuários. Neste caso a sessão é o líder, se kill na sessão, kill em todos os processos filhos.
- ❑ l-is multi-threaded – processos multiencadeados

```
ubuntu@eduardo-server: ~$ ps -aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.3	0.6	102208	13068	?	Ss	18:36	0:03	/sbin/init
root	2	0.0	0.0	0	0	?	S	18:36	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	18:36	0:00	[rcu_gp]
root	4	0.0	0.0	0	0	?	I<	18:36	0:00	[rcu_par_gp]
root	5	0.0	0.0	0	0	?	I<	18:36	0:00	[slub_flushwq]
root	6	0.0	0.0	0	0	?	I<	18:36	0:00	[netns]

- ❑ %CPU - Quanto da CPU o processo está usando
- ❑ %MEM - Quanta memória o processo está usando
- ❑ ADDR - Endereço de memória do processo
- ❑ C ou CP - Informações de uso e agendamento da CPU
- ❑ CMD - Nome do processo, incluindo argumentos, se houver
- ❑ NI - nice
- ❑ PID - Número de identificação do processo
- ❑ PPID - Número de identificação do processo pai do processo

```
ubuntu@eduardo-server: ~$ ps -aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.3	0.6	102208	13068	?	Ss	18:36	0:03	/sbin/init
root	2	0.0	0.0	0	0	?	S	18:36	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	18:36	0:00	[rcu_gp]
root	4	0.0	0.0	0	0	?	I<	18:36	0:00	[rcu_par_gp]
root	5	0.0	0.0	0	0	?	I<	18:36	0:00	[slub_flushwq]
root	6	0.0	0.0	0	0	?	I<	18:36	0:00	[netns]

- ❑ PRI - Prioridade do processo
- ❑ TIME - tempo de uso total da CPU
- ❑ TT ou TTY - Terminal associado ao processo
- ❑ WCHAN (Waiting Channel) - (rotina do kernel para processo waiting relativo ao endereço de memória) Endereço de memória do evento pelo

qual o processo está aguardando, processos em execução são marcados por um hífen (não há endereço fixo na memória para indicar)

Agradeço
a sua atenção!

Marcio Santana

marcio.santana@sptech.school

SÃO
PAULO
TECH
SCHOOL