

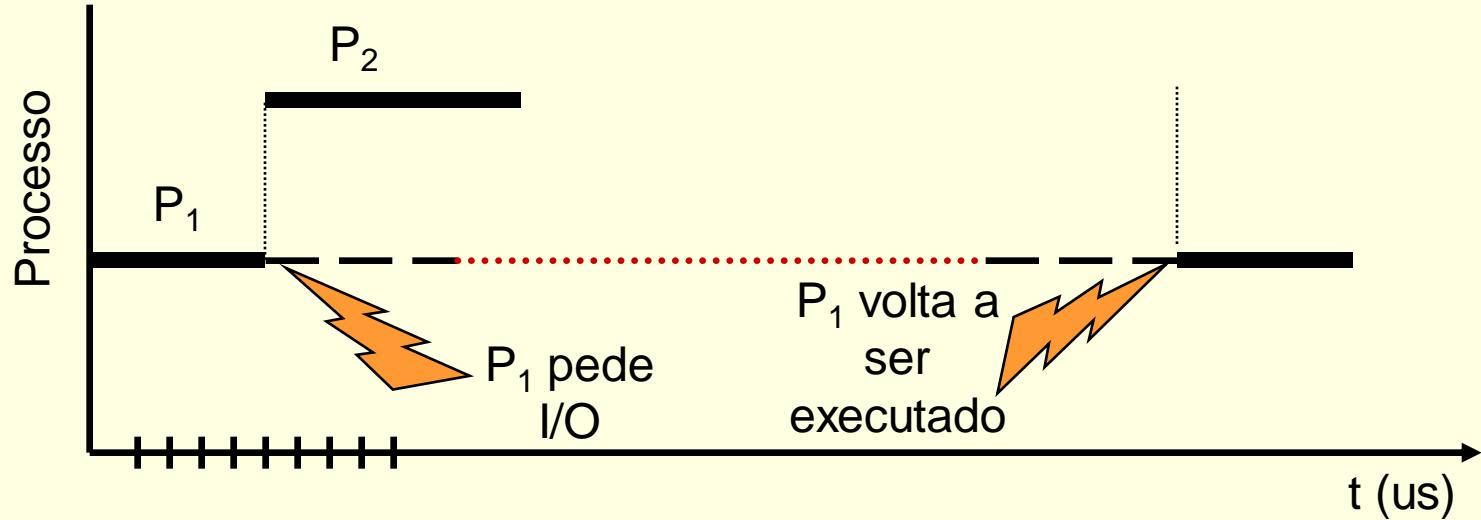
Fundamentos de Sistemas de Operação

Unix Windows NT Netware Mac OS DOS/V/S Vax/VMS
Linux Solaris HP/UX AIX Mach Chorus

Gestão de Processador
Multiprogramação

(Recordando...)

Comutação de Processos desencadeada por I/O



- Quando o processo P_1 pede uma operação de I/O, o SO escolhe um outro processo, P_2 , para executar.
 - Note-se que a escala do tempo de espera de I/O é cerca de mil vezes superior à escala do tempo usada (ms em vez de μs)

(Recordando...)

“Tipos de SOs”

Unix Windows NT Netware Mac OS DOS/V/S Vax/VMS Linux Solaris HP/UX AIX Mach Chorus

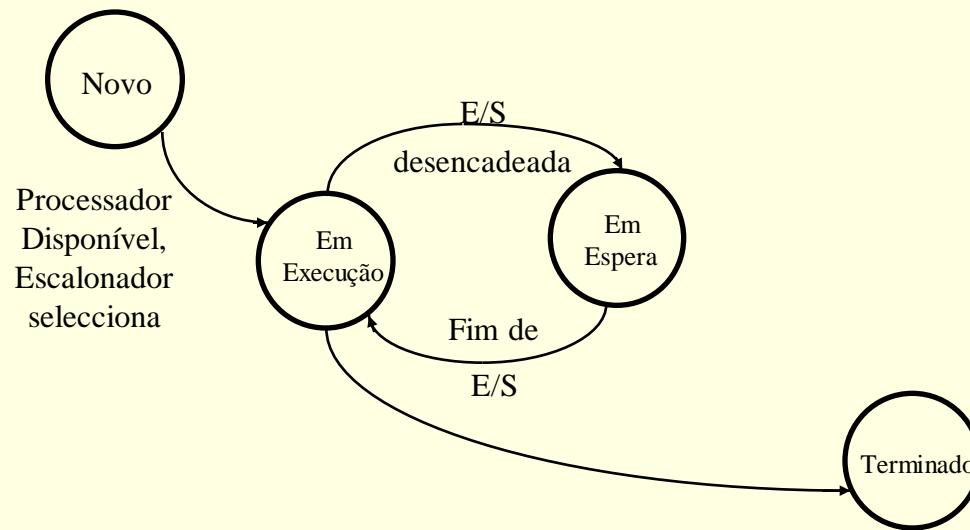
□ Evolução dos SOs (estas designações caíram em desuso)

- **Batch:** Executam *jobs* (cadeias de processos que não interagem com os utilizadores). Já não são usados. Contudo, os grandes supercomputadores (ver www.top500.org) são utilizados desta forma: a) o utilizador (p.ex. num portal) faz upload dum ficheiro que descreve os recursos de que necessita (CPUs, tempo esperado de execução, o(s) programa(s) a executar, os dados, etc.); b) o pedido entra numa fila de espera; c) “um dia” ☺ será processado; d) os resultados serão disponibilizados em ficheiros que o utilizador pode descarregar...
- **Timesharing:** Executam *tasks* (ou simplesmente processos). Como são usualmente acedidos de forma interactiva (via ecrã directamente ligado ou terminal remoto) são também designados **Interactivos** (por oposição aos Batch).

(Recordando...)

Comutação de Processos desencadeada por I/O

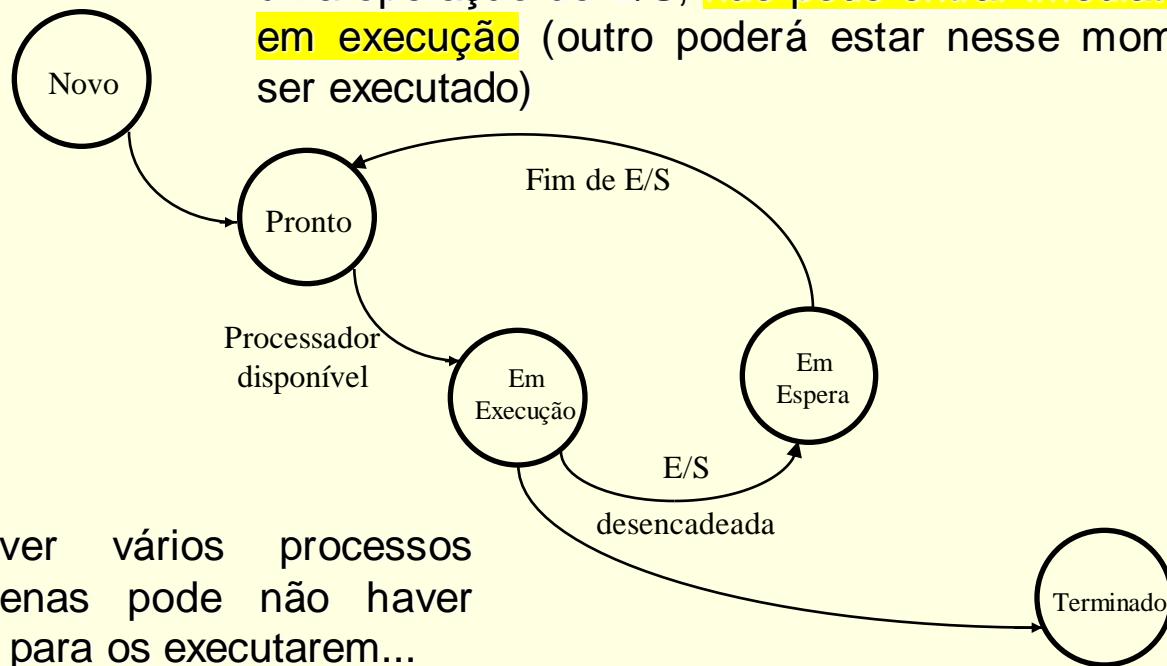
- Diagrama **muito simplificado** de estados e transições de um Processo:



Comutação de Processos desencadeada por I/O

- Diagrama mais detalhado:

De facto, um processo novo, ou um que viu acabar uma operação de E/S, **não pode entrar imediatamente em execução** (outro poderá estar nesse momento a ser executado)

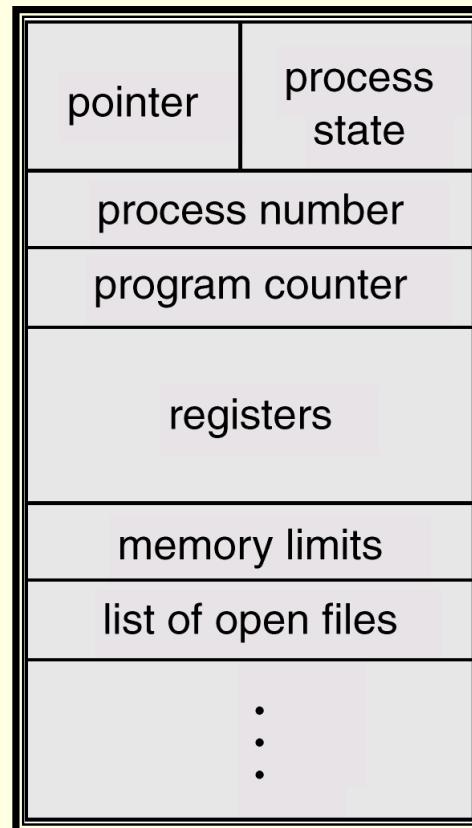


Poderá haver vários processos prontos; apenas pode não haver vários CPUs para os executarem...

Comutação de Processos: Estruturas de Dados

- *Process Control Block*: uma estrutura de dados para guardar informações que preservem o estado do processo de forma a que, se interrompida, posteriormente se possa retomar a sua execução.
- Conteúdo do PCB:
 - PC (ou IP): para retomar a sequência de instruções
 - SP: para recuperar o *stack*
 - Registo de estado: para recuperar as *flags*
 - Registros genéricos / Acumuladores
 - Informação para GM: RB/RL ou PTBR/PTLR (a ver mais tarde)
 - Outras ... nomeadamente um ID para o processo/PCB, informação para contabilização de recursos, sobre o estado do processo, sobre os ficheiros abertos...

O Process Control Block



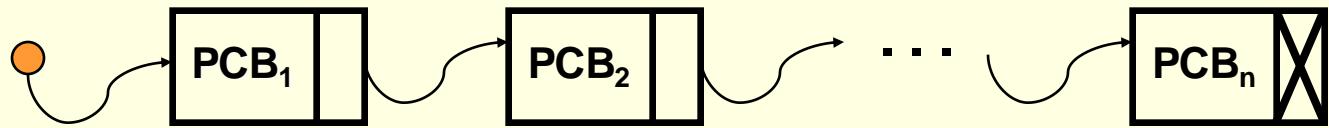
O Process Control Block... mais detalhado

Process management Registers Program counter Program status word Stack pointer Process state Priority Scheduling parameters Process ID Parent process Process group Signals Time when process started CPU time used Children's CPU time Time of next alarm	Memory management Pointer to text segment Pointer to data segment Pointer to stack segment	File management Root directory Working directory File descriptors User ID Group ID
--	--	--

A ver mais tarde...

Fila dos Processos Prontos: Como organizá-la?

- **Fila Ready:** a ordem dos processos no acesso ao CPU é a ordem por que se encontram os respectivos PCBs. [Esta é uma possível implementação para a fila de espera, não a única]



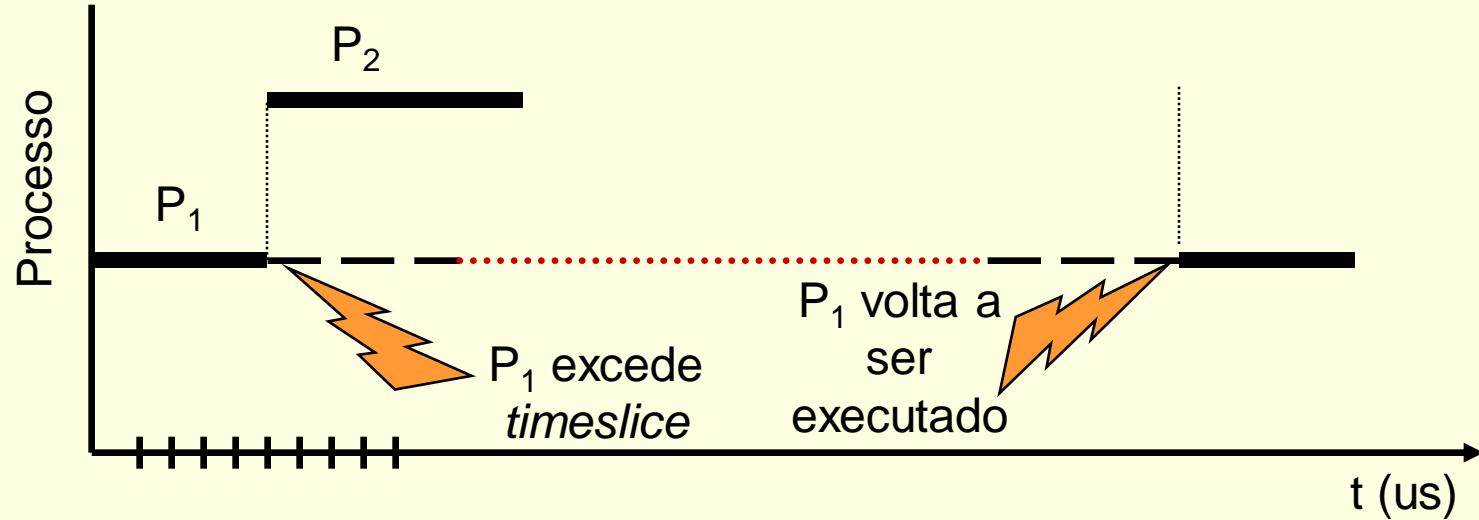
- O escalonador de médio prazo (MTS, *medium term scheduler*) implementa uma política de ordenação e, portanto, mantém os processos ordenados de acordo com essa política; à cabeça da fila está o “próximo a ser executado”. Move os “processos” da fila *waiting* para a *ready*, e intervém em situações de escassez de memória (pede à GM ...)

Fila dos Processos à Espera

□ Fila de Espera:

- Um PCB vai para a fila de espera quando o “seu processo” está à espera de um evento – que pode ser: fim de uma operação de I/O, fim de uma temporização (sleep), espera por um recurso,...
- O MTS move o PCB para a fila *ready* quando termina a situação de espera.

Comutação de Processos por preempção



- Quando um processo P_1 excede a sua fatia de tempo (*timeslice*) de execução, há um *interrupt*, o SO é chamado e escolhe um (talvez outro) processo, P_2 , para executar.

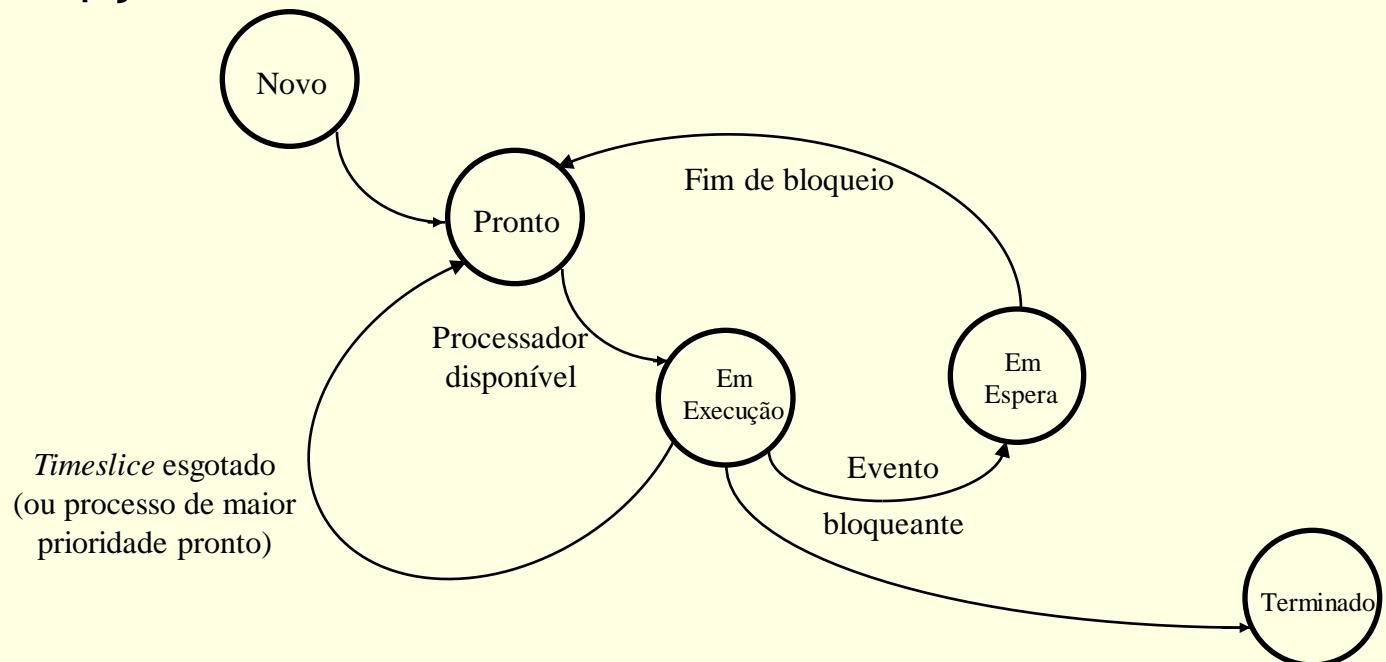
Comutação de Processos por preempção

□ Implementação da fatia de tempo:

- O SO (escalonador) programa (carrega) um contador *hardware* com um valor cuja magnitude representa a duração da *timeslice*; o contador vai sendo automaticamente decrementado ...
- Se o contador chega a zero provoca uma interrupção que activa o escalonador de curto prazo, e este “remove” o processo do CPU (ver slide 21)
- Se o processo “abandona voluntariamente” o CPU (por espera) quando regressar à execução o contador é re-carregado de novo com a) o tempo que lhe restava, ou b) o valor inicial... (depende do grau de “sofisticação” do SO).

Comutação de Processos

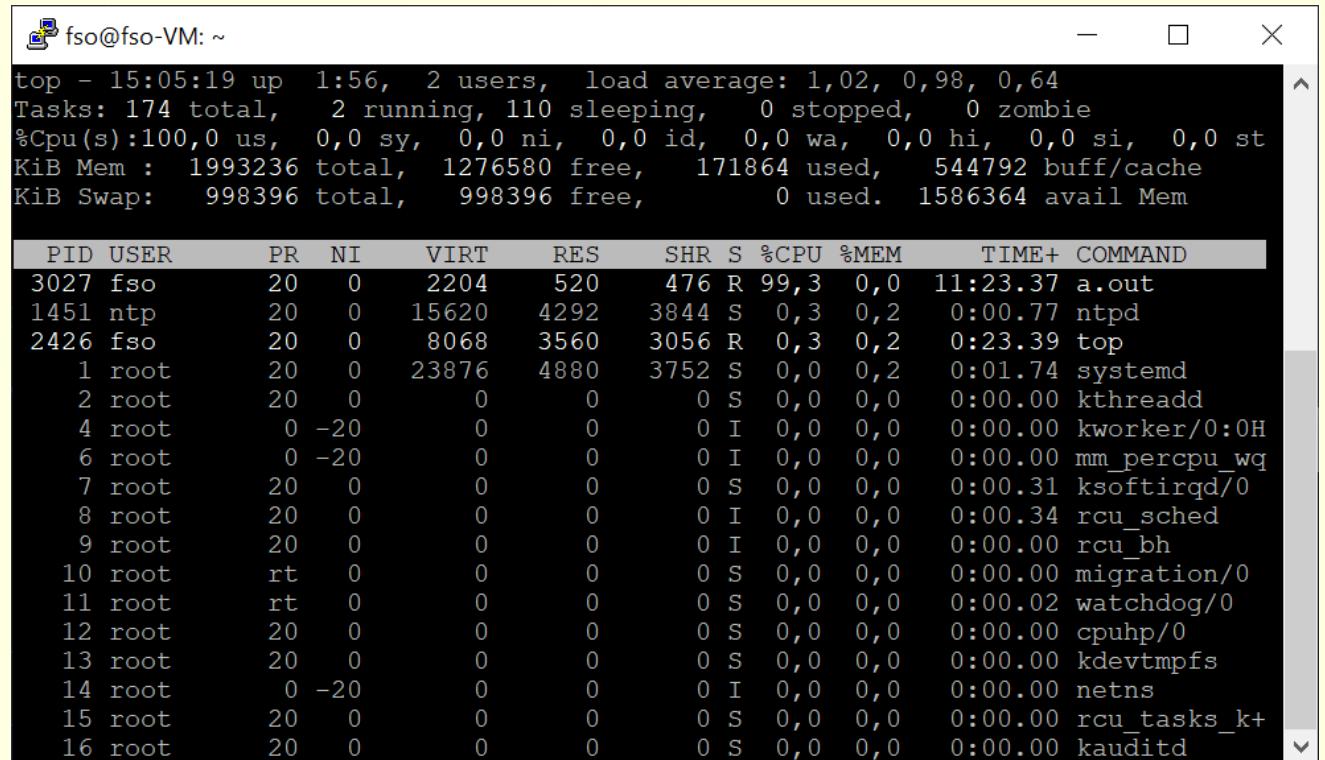
- Diagrama de estados e transições completo para SO's que comutam processos por I/O, ou outro evento bloqueante, e por preempção:



Comutação de Processos

□ Demo Linux:

- Estados (+importantes) dos processos visíveis no top: (R)unning, (S)eeping



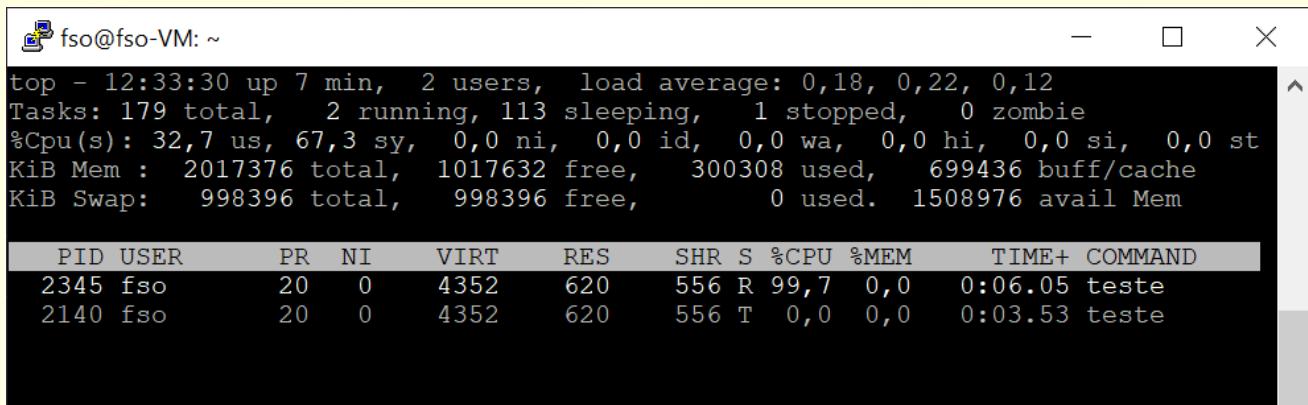
The screenshot shows a terminal window titled 'fso@fso-VM: ~'. The window displays the output of the 'top' command. The top part of the output shows system statistics: tasks (174 total, 2 running, 110 sleeping, 0 stopped, 0 zombie), CPU usage (%Cpu(s): 100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st), memory usage (KiB Mem: 1993236 total, 1276580 free, 171864 used, 544792 buff/cache, KiB Swap: 998396 total, 998396 free, 0 used, 1586364 avail Mem). The bottom part of the output is a table showing the top 16 processes:

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3027	fso	20	0	2204	520	476	R	99,3	0,0	11:23.37	a.out
1451	ntp	20	0	15620	4292	3844	S	0,3	0,2	0:00.77	ntpd
2426	fso	20	0	8068	3560	3056	R	0,3	0,2	0:23.39	top
1	root	20	0	23876	4880	3752	S	0,0	0,2	0:01.74	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0,0	0,0	0:00.31	ksoftirqd/0
8	root	20	0	0	0	0	I	0,0	0,0	0:00.34	rcu_sched
9	root	20	0	0	0	0	I	0,0	0,0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0,0	0,0	0:00.00	migration/0
11	root	rt	0	0	0	0	S	0,0	0,0	0:00.02	watchdog/0
12	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kdevtmpfs
14	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	netns
15	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_tasks_k+
16	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kauditd

Comutação de Processos

□ Demo Linux:

- **Estado S(T)opped:** muitas vezes os estudantes querem “matar” um processo e “fazem um disparate” 😊: carregam em CTRL-Z. Isso não termina o processo, coloca-o no estado stopped e liberta o terminal... 😊



fso@fso-VM: ~

```
top - 12:33:30 up 7 min,  2 users,  load average: 0,18, 0,22, 0,12
Tasks: 179 total,   2 running, 113 sleeping,   1 stopped,   0 zombie
%Cpu(s): 32,7 us, 67,3 sy,  0,0 ni,  0,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem : 2017376 total, 1017632 free, 300308 used, 699436 buff/cache
KiB Swap: 998396 total, 998396 free,      0 used. 1508976 avail Mem

      PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
  2345 fso        20   0    4352    620    556 R 99,7  0,0   0:06.05 teste
  2140 fso        20   0    4352    620    556 T  0,0  0,0   0:03.53 teste
```

- Se quiserem “recuperar” o processo e querem que continue a correr, fazer (na bash): \$ **bg**
- Se quiserem matá-lo, neste caso seria: \$ **kill 2140** (ou **kill -9 2140**)

Comutação de Processos

□ Desvantagens:

- A comutação de processos leva tempo, não é instantânea; o tempo gasto na comutação não é “produtivo” (é *overhead*) ...
- Na implementação de um SO o tempo gasto para comutar de um processo para outro deve ser minimizado, e é designado latência do dispatcher
- Minimizar o número de comutações (*timeslices* grandes) maximiza o uso produtivo do CPU (a correr aplicações!)... mas piora o tempo de resposta para os processos “interactivos”...
- Os SOs permitem configurar o *timeslice* de forma a favorecer os “interactivos” ou os *CPU-bound* ...

Comutação de Processos: desafio

- Tente imaginar uma forma simples...
 - ... de estimar (ainda que muito grosseiramente) o tempo gasto na Comutação de Processos [Nota: pretende-se que tenha uma ideia de como fazer, não que escreva o código]