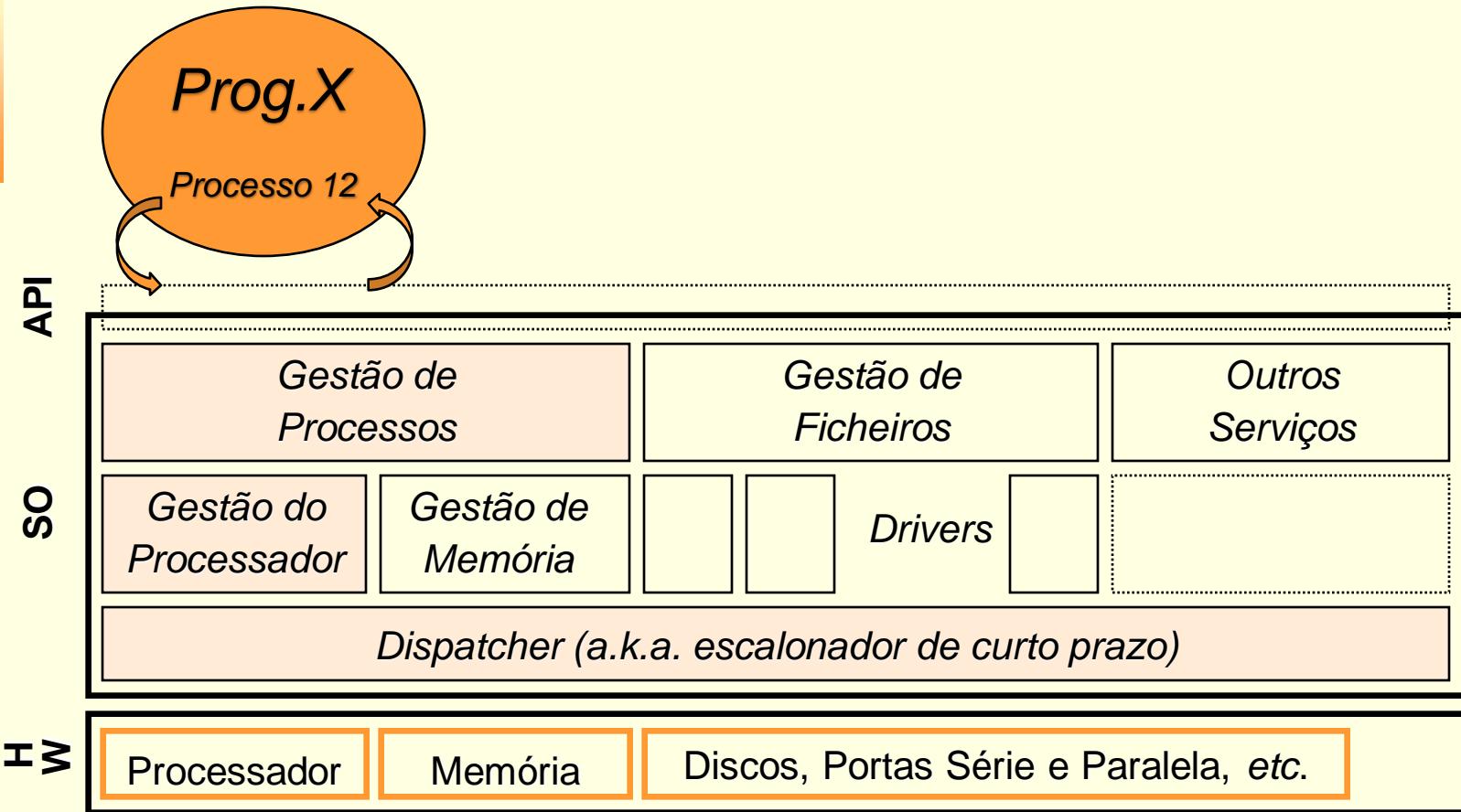


Fundamentos de Sistemas de Operação

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

Gestão de Processos:
Executar, ..., executar, até terminar?

O SO e os processos



SO: Gestão de Processos

□ Iniciar a execução

- “Agora” que o SO já criou o EE, definiu nele as regiões (código, dados, etc.) e carregou nestas o que havia a carregar (e protegeu o que havia a proteger ☺), como fazer para correr o programa?
- [Nota: podemos dizer que os registos do processador se enquadram em 3 grupos: gerais (EAX, EBX, ... EIP, ESP, ...), controle (CR0 a CR4) e “memória” (GDTR, ...) – veremos mais tarde...]
- O SO carrega numa estrutura de dados (“contexto”) os valores que devem ser colocados nos registos do processador extraídos do ficheiro executável - nos gerais como EBX, um “valor”, no EIP, o endereço da 1^a instrução do programa, nos “da memória” as informações sobre a colocação do EE em RAM ...

(continua)

SO: Gestão de Processos

□ Iniciar a execução

(continuação)

- Depois num “passe de mágica” ☺ [nota: é das poucas partes do SO escritas em assembly, não seria possível fazê-lo de outra forma] o SO coloca o contexto no processador, **o que faz com que o processo comece a correr...**
- Os módulos do SO que desempenham estas tarefas são, na figura que desenhamos, a “Gestão do Processador” e o “dispatcher”... **[NOTA IMPORTANTE]**: as “bonitas” figuras da estrutura modular do SO que nos têm acompanhado são uma “visão de artista” – nada, no mundo real dos SOs é assim “tão perfeitinho”, “tão estruturadinho”]
- Veremos mais adiante que o contexto é parte de uma estrutura mais complexa, o Process Control Block (PCB) que tem outras informações sobre o processo...

SO: Gestão de Processos

□ Taxa de utilização do processador

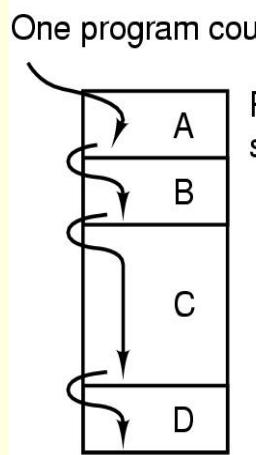
- Se o SO apenas permitisse correr um único processo de cada vez, a taxa de utilização do processador seria baixíssima; senão vejamos:
 - Um processador moderno tem um clock de, pelo menos, 1 GHz, i.e., a capacidade de executar 1 instrução em cada nano-segundo (ns)
 - Um disco SSD muito rápido é capaz de ler um bloco em 100 microsegundos (um HDD pode ser 100x mais lento ☺). Portanto, enquanto o processo está à espera da leitura de 1 bloco, poderia ter executado 100000 instruções (recorde-se de AC: estamos a ser muito optimistas e considerar que está tudo nas caches L1 do CPU)
- Para melhorar a taxa de uso do CPU é preciso correr outro processo quando “este” está à espera do disco...

SO: Gestão de Processos

□ SOs multiprogramação

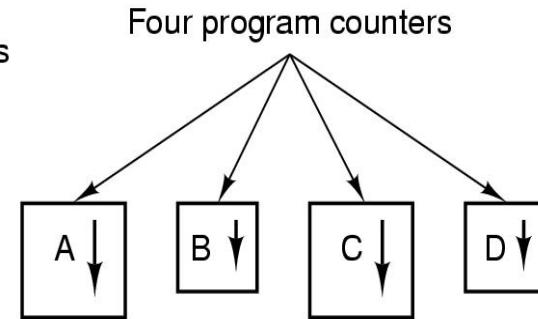
- Um SO capaz de correr concorrentemente múltiplos processos designa-se de multiprogramação. Para o conseguir,
 - A GM tem de ser capaz de gerir múltiplos EE carregados em RAM
 - Cada processo tem uma estrutura de dados (PCB – Process Control Block) que inclui o “contexto” e outras informações
 - O dispatcher é capaz de trocar de contexto entre dois processos: um é “retirado” de execução e outro colocado em execução
- Como é que se pode correr simultaneamente vários processos se só houver 1 processador?

O Modelo para os Processos

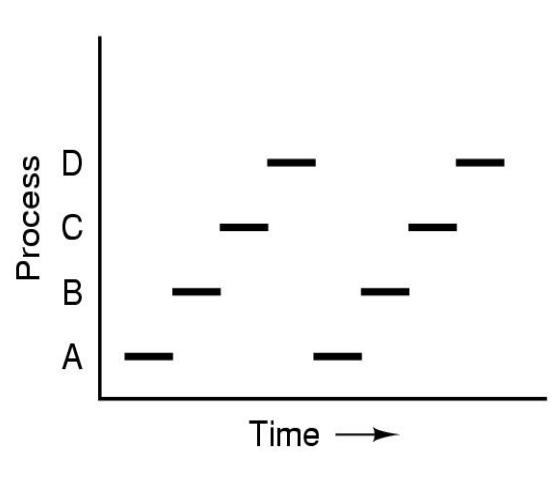


(a)

Process
switch



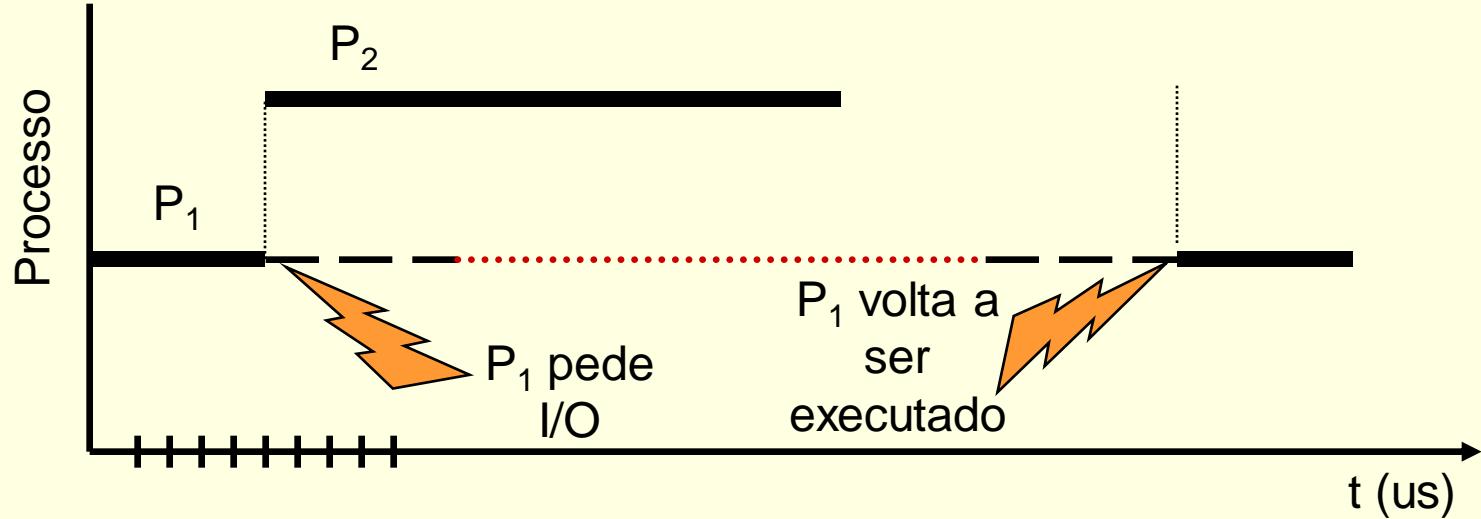
(b)



(c)

- Podemos ver a execução “simultânea” de múltiplos processos de diferentes formas: **a)** Vários processos carregados em memória, e um PC (contexto) que “salta” de um processo para outro; **b)** Um conjunto de processos sequenciais independentes, cada um com o seu próprio PC (contexto); **c)** Uma sequência “entremeada”, com apenas um processo activo de cada vez

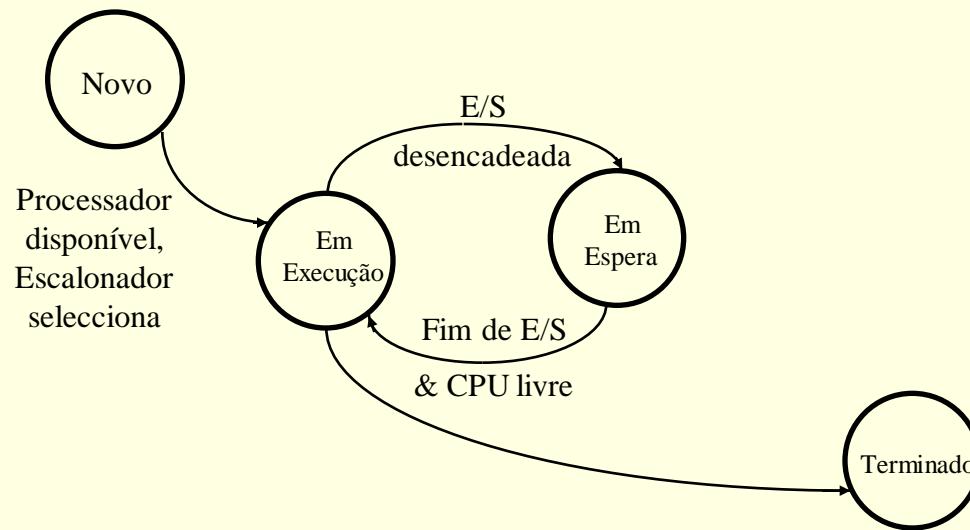
Comutação de Processos desencadeada por I/O



- Quando o processo P₁ pede uma operação de I/O, o SO escolhe um outro processo, P₂, para executar.
 - Note-se que a escala do tempo de espera de I/O é cerca de mil vezes superior à escala do tempo usado (ms em vez de μ s)

Comutação de Processos desencadeada por I/O

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



Estados de um Processo

- Durante a execução, o processo vai mudando de estado:
 - **Novo:** Quando o processo é criado.
 - **Em execução:** Quando as suas instruções estão a ser executadas.
 - **Em espera, ou Bloqueado:** o processo está à espera que ocorra um dado evento – neste caso, que a E/S termine.
 - **Terminado:** o processo terminou a execução do “seu” programa.

Comutação de Processos desencadeada por I/O

□ Vantagens:

- **Máxima ocupação do CPU:** Um processo corre “produtivamente” até precisar de I/O.
- Bom para sistemas *batch* (veremos mais tarde os diferentes “tipos de sistemas”, mas para já estes são sistemas com os quais as únicas interacções que os utilizadores têm é para pedir a execução de um “trabalho” e depois, talvez horas ou até dias depois, irem buscar os resultados – é assim que se usam os grandes “supercomputadores” – vejam www.top500.org)

□ Desvantagens:

- **Run-away process:** um processo que entra em *loop* sem fazer I/O monopoliza o CPU – “nunca” mais o larga...
- Mau para sistemas interactivos (e.g., laptops Windows ou Linux)