Fundamentos de Sistemas de Operação

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

Programas e Processos:

Introdução

Resumo da aula anterior…

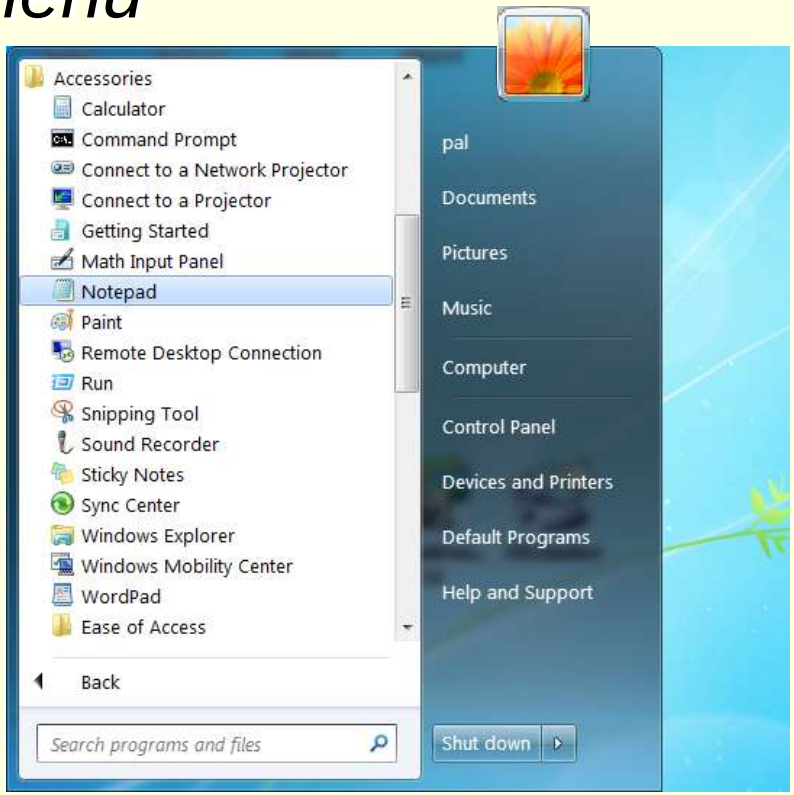
- *O que sabemos (recordamos):*
  - *Um processador executa programas em linguagem-máquina*
  - *Um programa, para ser executado tem de estar carregado em memória (RAM)*

  e ainda que
  - *Os programas são inicialmente armazenados em dispositivos de memória persistente – discos, memórias flash – sob a forma de ficheiros*

- *O que ☺ intuitivamente ☺ sabemos:*
  - *A gestão dos ficheiros em disco é tarefa do SO*
  - *Para executar um programa interagimos com o SO (click, enter, comando…)*
  - *Para carregar um programa em memória é preciso que haja espaço livre…*

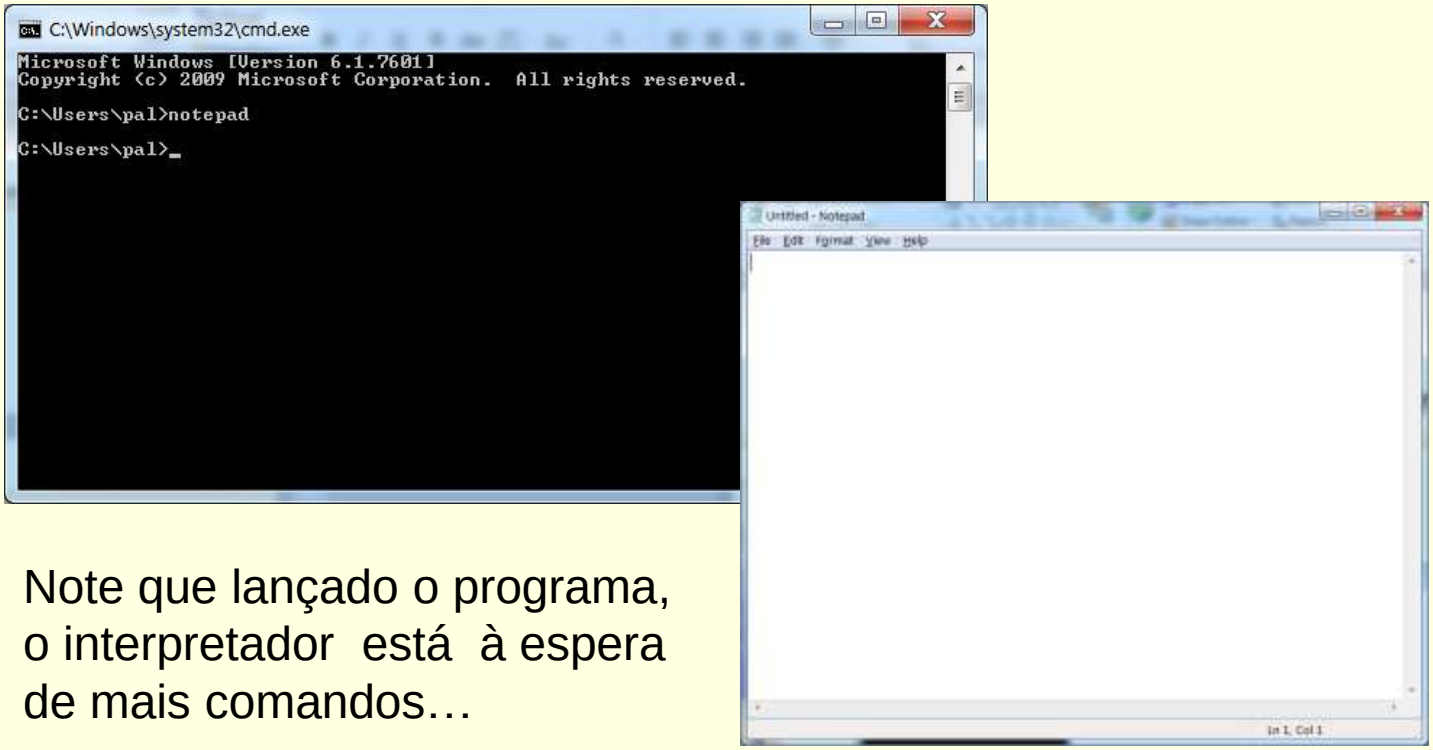
Executando um novo programa (1)

- *Navegação em menu*



Executando um novo programa (2)

- *Interpretador de comandos*



Note que lançado o programa, o interpretador está à espera de mais comandos…

Executando um novo programa (3)

- *Interagindo com "o sistema"*
  - *Usando o mouse e clicks*
    - *Por duplo click no ícone do programa, ou navegando nos menus "top-down"*

  ou
  - *Usando a janela de comandos*
- *O que ☺ intuitivamente ☺ sabemos:*
  - *A nossa escolha/comando determina o (ficheiro) programa executável*
  - *O "sistema" localiza o ficheiro e carrega o seu conteúdo para memória*
  - *A execução do programa há de começar no ☺ endereço certo ☺…*
  - *Usualmente é criada uma janela para o novo programa*
- *TPC: experimente no Linux*

Executando um novo programa (4)

- *E se quisermos que um programa lance outro?*
  - *Temos de incluir no código um "pedido de serviço" ao SO*

```
int main(…)
{
   ...
   ... // pedido de serviço...
   ... // e aqui já estará um novo processo a correr...
   ...
}
```

  - *A função* **`fork()`** *é o passo fundamental, nos sistemas operativos compatíveis POSIX (de momento pensemos que este termo se refere ao Unix e "derivados", como o Linux) para executar um novo programa.*
  - *No Windows, o mesmo resultado consegue-se com a função* **`CreateProcess(arg1,…arg10)`**

Programas de sistema

- *Interpretador de comandos*
  - *Será que podemos escrever o nosso próprio interpretador de comandos?*
    - *Sim; e também o nosso próprio editor de texto, etc...*
- *Programas de sistema e Utilitários*
  - *Os programas "críticos" para o funcionamento do sistema – como os interpretadores de comandos – e.g.,* **`cmd.com`** *no Windows e* **`shell`** *no Linux, são designados programas (ou aplicações) de sistema (system applications).*
    - *Outros programas usualmente classificados como "de sistema" são: compiladores, assemblers, linkers, etc… mas esta classificação não é rigorosa.*
  - *São chamados utilitários programas usados para desempenhar tarefas necessárias ao "bom funcionamento" do sistema – e.g., editor de texto, programas de cópia (backup) de ficheiros, etc.*

Algumas definições/conceitos… *ainda que não totalmente refinados* (1)

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

- *Programa executável*
  - *Ficheiro com um formato predefinido pelo SO, contendo zonas de dados e instruções-máquina reconhecidas pelo CPU onde vai ser executado. Entidade passiva*
- *Processo*
  - *Programa em execução; num dado instante, caracteriza-se pelo seu "estado" e ocupação de recursos do sistema: físicos, ou hardware – CPU, memória – e lógicos – e.g., canais de comunicação com o utilizador. Entidade activa: muda de "instante para instante"… até terminar ☺*

Algumas definições/conceitos… *ainda que não totalmente refinados* (2)

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

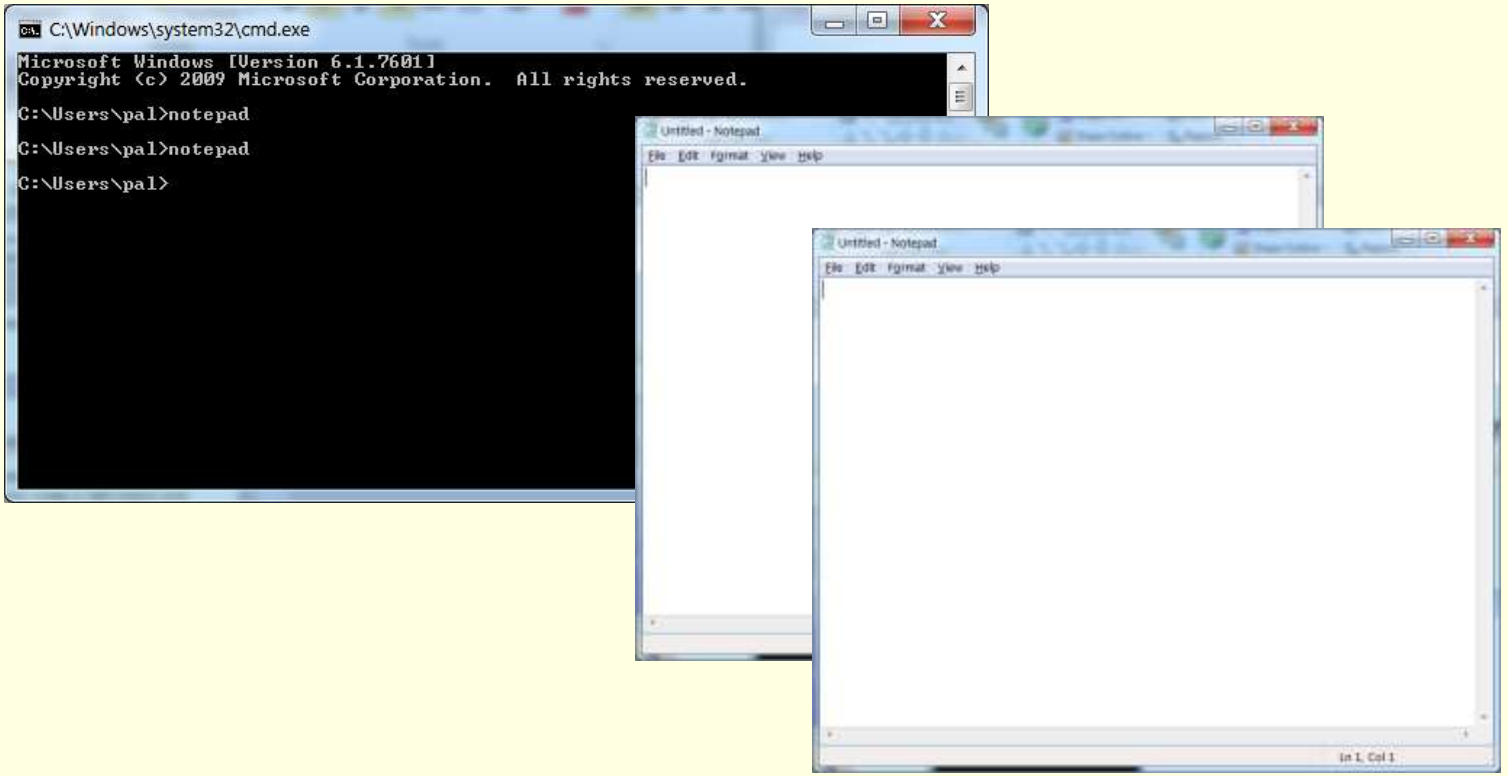
- *Sistema de Operação*
  - *O SO é um prestador de serviços - às aplicações - e, simultaneamente, um gestor de recursos: CPU, memória, periféricos. O SO oferece um leque de abstracções (e.g., ficheiros, canais de comunicação) que são cruciais para simplificar o desenvolvimento e suportar a execução das aplicações.*
  - *Pode dizer-se que o SO oferece à aplicação uma máquina virtual "de nível superior" ou, se quisermos, "mais adequada" - que a máquina hardware básica - para suportar as necessidades das aplicações.*
    - *Por exemplo, o SO pode "virtualizar" uma máquina abstracta independente para cada processo aplicacional que executa, de forma que a aplicação não tenha de gerir ocupações de memória, tempos de CPU, decidir quando pode aceder a um periférico que também é acedido por outra aplicação, etc.*

Algumas definições/conceitos… ainda que não totalmente refinados (3)

- *Núcleo vs. Programas de sistema*
  - *Já sabemos que os programas de sistema (interpretadores de comandos - gráficos ou não), utilitários (editor, utilitários de backup) etc., são aplicações "como as outras". É no núcleo do SO que são realizadas as abstracções e executados os serviços tão necessários às aplicações.*
  - *Finalmente, resta saber como é que uma aplicação pode solicitar ao SO os tão propalados serviços que este oferece.*
    - *Do ponto de vista da programação, tal faz-se invocando funções que fazem "chamadas ao sistema" (system calls).*
    - *Do ponto de vista da execução-máquina de tais chamadas, esta faz-se de forma algo parecida com a chamada de uma subrotina, no sentido em que: há uma salvaguarda do estado do CPU; há um salto do PC (IP) para um endereço destino; há uma execução de algum código; e há um retorno ao "ponto de origem" com reposição do estado do CPU. [A desenvolver ☺]*

Para pensar… (1)

- *Dois processos… um mesmo programa!*



Para pensar … (2)

- *Os dois processos* (notepad)…
  - *Podem estar a ser simultaneamente em execução? Ou quando um está activo o outro está, obrigatoriamente, suspenso?*
  - *Ocupam a mesma zona de memória (RAM)?*

Para experimentar…

- *Interacção utilizador/SO*
  - *No Windows, como se vê quais os processos que estão a correr? E no Linux?*
  - *Tente abrir uma "janela (interpretador) de comandos" no Linux e lançar duas vezes o editor (gedit). O comportamento é idêntico ao mostrado no slide 11?*

Fundamentos de Sistemas de Operação

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

Programas, Processos, e o Sistema de Operação...

Carregar 2x o mesmo programa em memória?


```
…
08048374 <main>:
 8048374: 55                                  push   %ebp
 8048375: 89 e5                               mov    %esp,%ebp
 8048377: 83 ec 08                            sub    $0x8,%esp
 804837a: 83 e4 f0                            and    $0xfffffff0,%esp
 804837d: b8 00 00 00 00                      mov    $0x0,%eax
 8048382: 83 c0 0f                            add    $0xf,%eax
 8048385: 83 c0 0f                            add    $0xf,%eax
 8048388: c1 e8 04                            shr    $0x4,%eax
 804838b: c1 e0 04                            shl    $0x4,%eax
 804838e: 29 c4                               sub    %eax,%esp
 8048390: c7 45 fc 01 00 00 00                movl   $0x1,0xfffffffc(%ebp)
 8048397: c7 05 dc 95 04 08 02                movl   $0x2,0x80495dc
 804839e: 00 00 00
 80483a1: 8b 45 f8                            mov    0xfffffff8(%ebp),%eax
 80483a4: c7 00 03 00 00 00                   movl   $0x3,(%eax)
 80483aa: c9                                  leave
 80483ab: c3                                  ret
```

A variável i é guardada no endereço ebp-4, que é um endereço no stack

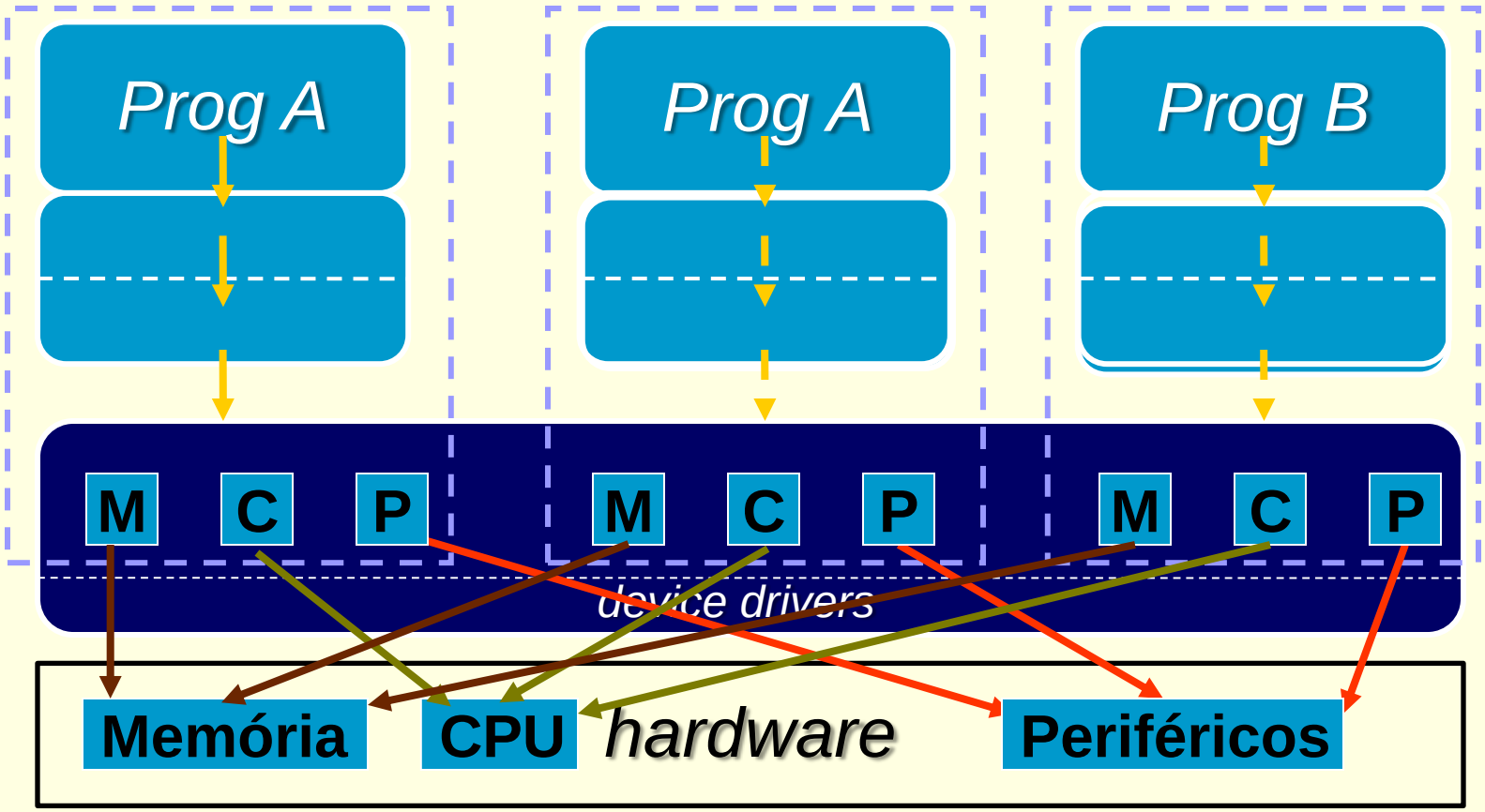
O código executável do programa é para ser carregado em memória a partir da posição 0x8048374

A variável x é guardada no endereço 0x80495dc

Respondendo ao "para pensar"… (do slide 12)

- *Os dois processos…*
  - *Podem estar a ser simultaneamente em execução: intuitivamente, esperamos isso de um sistema que tem um processador com vários cores…*
  - *Mas como é intuitivamente evidente, eles não podem ocupar a mesma zona de memória (RAM)*
- *Como resolver o problema?*

O processo como uma Máquina Virtual
que executa um programa



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

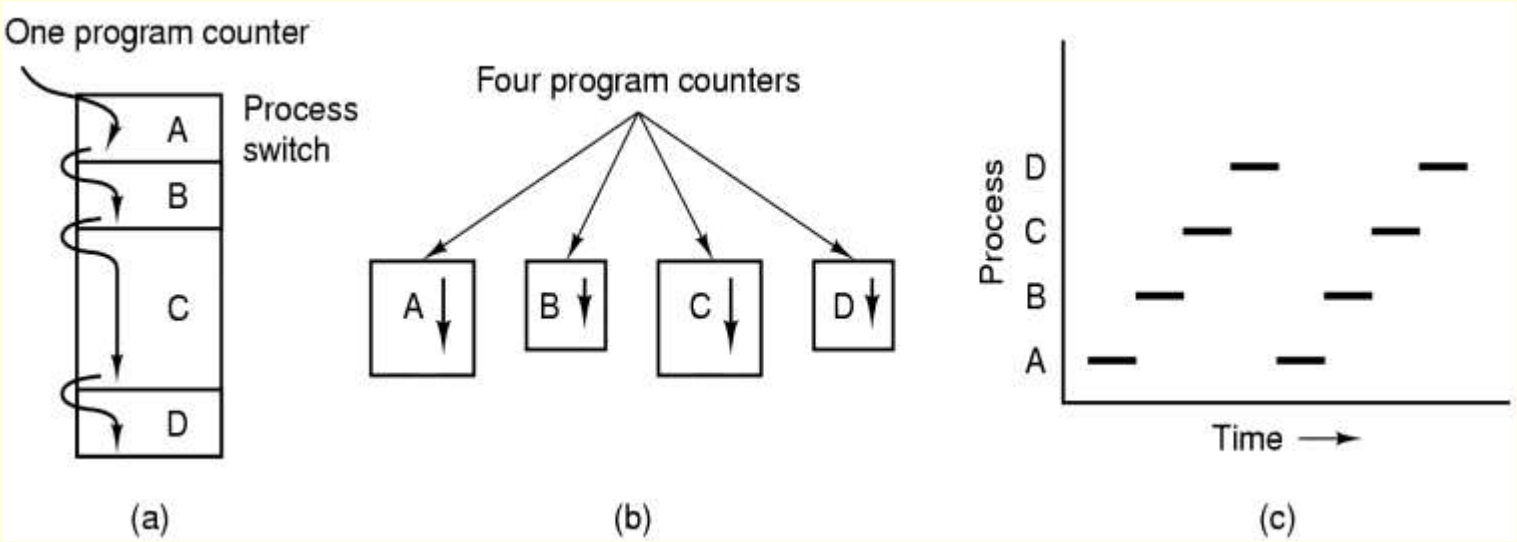
Respondendo ao "para pensar"…

- *O processo, conceito oferecido pelo SO,*
  - *"Simula um novo sistema" que tem um CPU, uma memória e periféricos idênticos aos da "máquina real". Diz-se que VIRTUALIZA esses recursos*
  - *Cada programa corre nesta máquina virtual como se mais nada existisse… por isso, quando uma cópia do programa A é carregada no endereço 0x8048374 e a outra cópia é carregada noutro processo exactamente no mesmo endereço, elas não se sobrepõem porque as MVs são distintas!*
  - *O SO encarrega-se de garantir que, na memória real, as zonas de memória "adjudicadas" a cada um dos processos (MVs) não se sobrepõem. Como faz isso? Mais tarde veremos… ☺*

Virtualização, particionamento ou partilha?

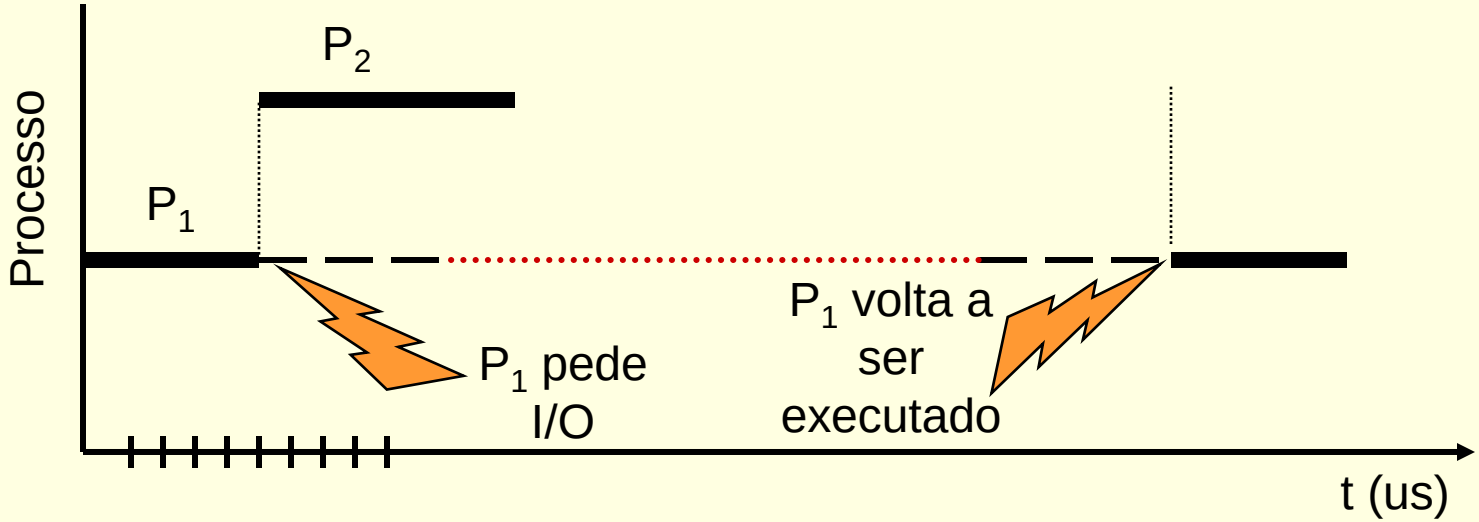
- *Consideremos, agora como "virtualizar" os recursos M, C e P*
  - *M- memória: uma técnica simples seria o SO "adjudicar" a cada processo a porção suficiente de RAM para "conter" o programa. Enquanto o "sistema real" tivesse RAM livre, não haveria problema…*
  - *C- CPU: ao contrário da RAM, o CPU (ou mais exactamente - nos tempos modernos - um core de um CPU; para resolver esta ambiguidade falaremos de processador) é indivisível; como poderemos ter um número de processadores virtuais superior ao número de processadores físicos?*
    - *A solução passa por atribuir processadores aos processos em "fatias de tempo", ou timeslices: um processador é "adjudicado" durante algum tempo a um processo, depois a outro, … , e assim sucessivamente, até eventualmente ser novamente adjudicado ao primeiro… (Esta técnica é melhor descrita como uma partilha do recurso – processador – do que como um particionamento…)*
  - *P – periféricos: não abordaremos para já este caso ☺*

Que Modelo para os Processos?



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

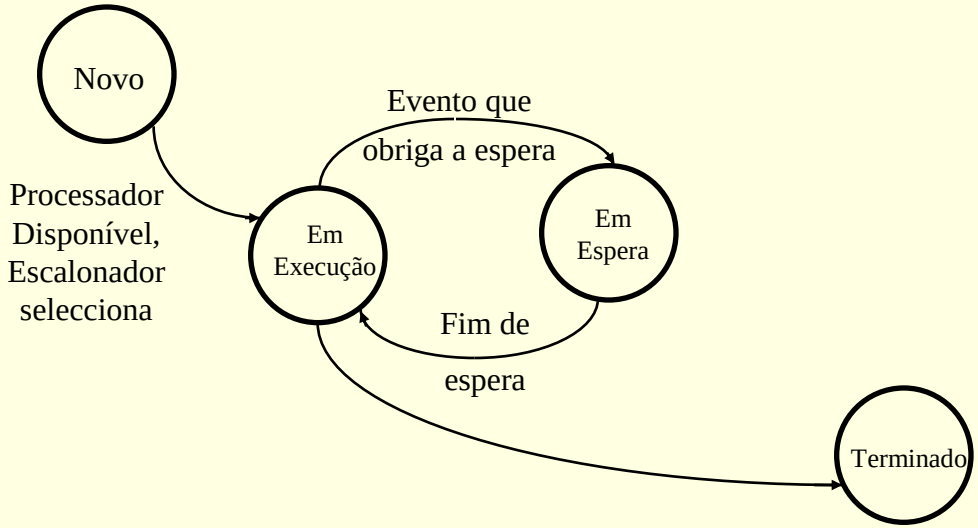
Comutação de Processos



- Quando o processo P_1 perde direito ao CPU, o SO escolhe um outro processo, P_2, para executar.
  - Note-se que se P_1 está, por ex., à espera de um dado que vem do disco a escala do tempo para a linha de I/O é cerca de mil vezes superior à escala do tempo usada no resto (ms em vez de us)

Estados de um Processo

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



Estados de um Processo

- *Durante a vida, um 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. Não precisa do CPU ☺*
  - ***Terminado:*** *o processo terminou a execução do "seu" programa.*

Atributos de um Processo

- *Os mais importantes são:*
  - ***PID:*** *Identifica inequivocamente o processo (process identifier).*
  - ***Utilizador:*** *Utilizador que criou o processo.*
  - ***Nome do executável:*** *nome do programa (ficheiro) executável que está a correr*
  - ***Data de criação:*** *data e hora em que o processo foi criado*
  - ***Informação para contabilização de consumos de recursos:*** *tempo de CPU usado até ao momento; quantidade de memória usada no instante da observação.*
  - ***Prioridade:*** *importância relativa do processo (face aos outros processos) quando este precisar do CPU*

Atributos de um Processo: TPC

- *Compare os atributos exibidos pelo*
  - *Windows Task Manager*
  - **`top`** *do Linux*
  - *Tente aprender um pouco sobre eles lendo o manual/help do programa, documentação on-line, etc…*

O que pode o SO fazer para/pelos processos?

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

- Um SO tem de oferecer, pelo menos, os seguintes serviços:
  - **Criar:** Criação de um novo processo, e identificação do programa que este vai executar.
  - **Suspender:** Suspender a execução de um processo de forma a que mais tarde a possamos continuar.
  - **Retomar:** Retomar a execução de um processo previamente suspenso
  - **Terminar:** fazer com que o processo termine a execução do "seu" programa.
  - **Consultar atributos:** obter valores dos atributos

SO: módulos fundamentais

- *Módulos (em destaque) cujas funções já começamos a compreender…*