Fundamentos de Sistemas de Operação

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

Parte II: Gestão de Processador

a) Multiprogramação

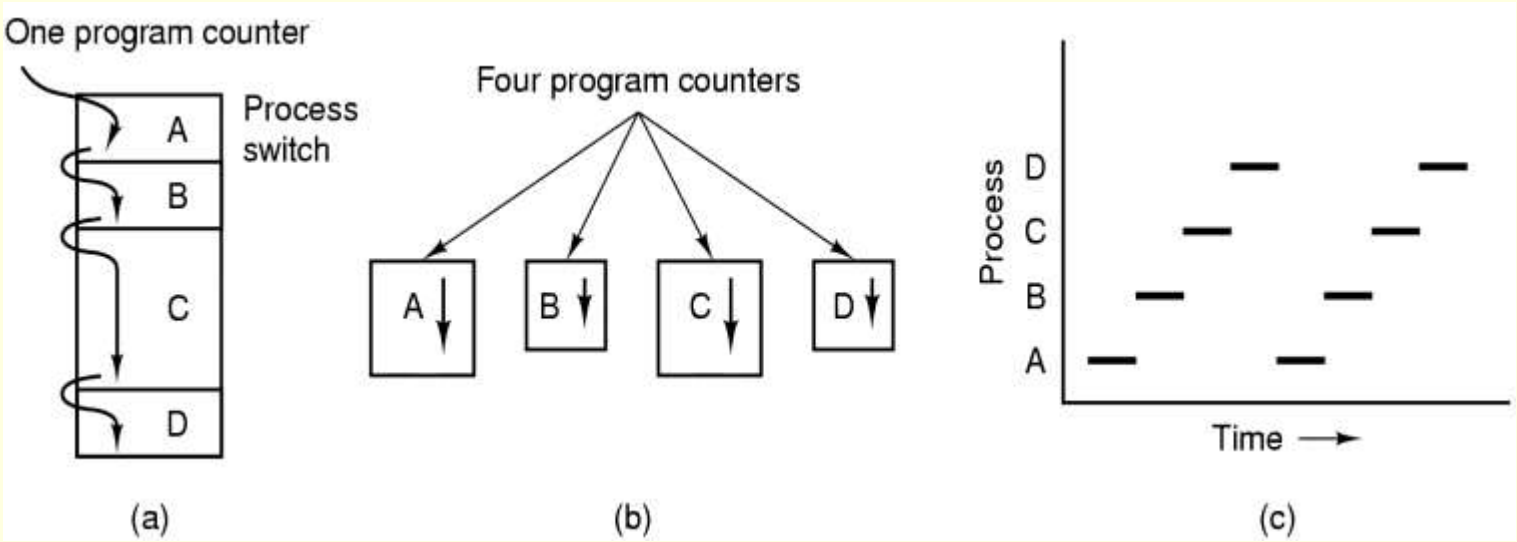
Processo: Relembrando Conceitos…

- Processo: programa em execução; esta progride de forma sequencial.
- Tipos de SOs:
  - **Batch:** Executam *jobs* (processos não interactivos).
  - **Interactivos, ou *timesharing*:** Executam *tasks* (ou simplesmente processos).
  - De facto, processos, *tasks* e *jobs* podem ser considerados como sinónimos; contudo a terminologia usada num caso específico depende da "carga histórica", do SO concreto…

Processo: Relembrando Conceitos…

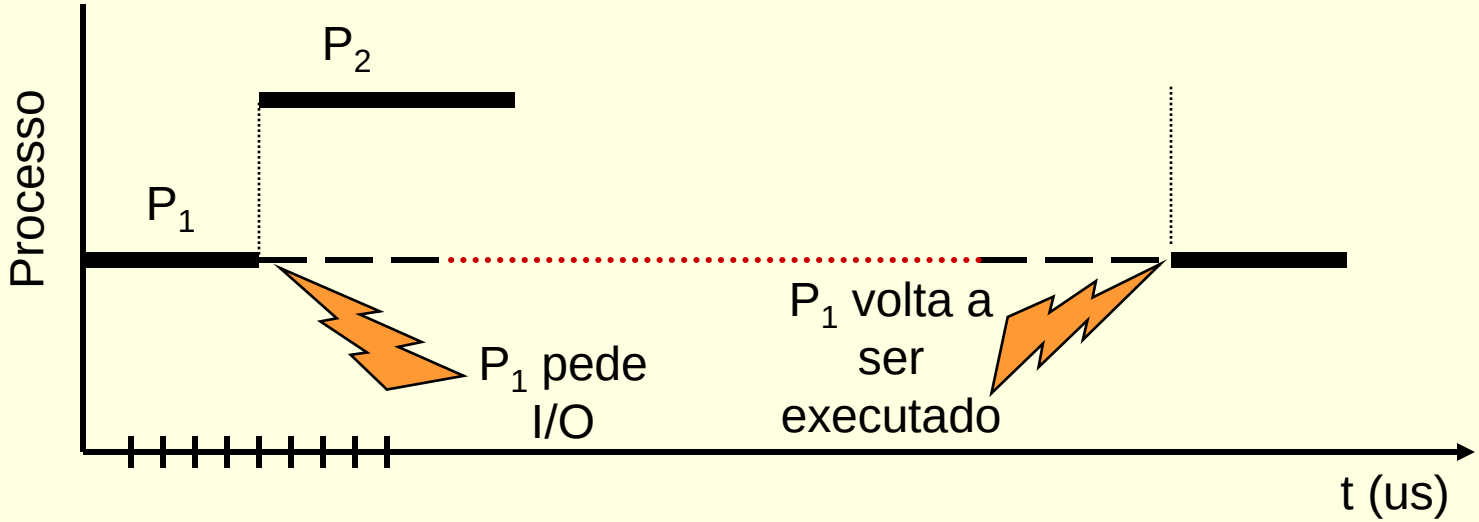
- Um processo inclui:
  - **Zona de código:** "percorrida" pelo *program counter*.
  - **Zona de Stack:** apontada pelo *stack pointer*.
  - **Zona de dados.**

O 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

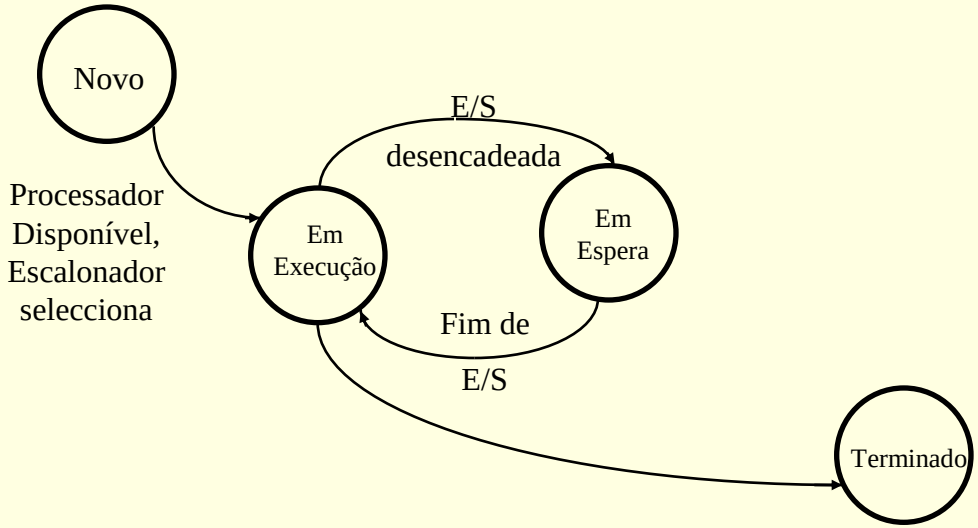
Comutação de Processos desencadeada por I/O



- 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 us)

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.
  - **Terminado:** o processo terminou a execução do "seu" programa.

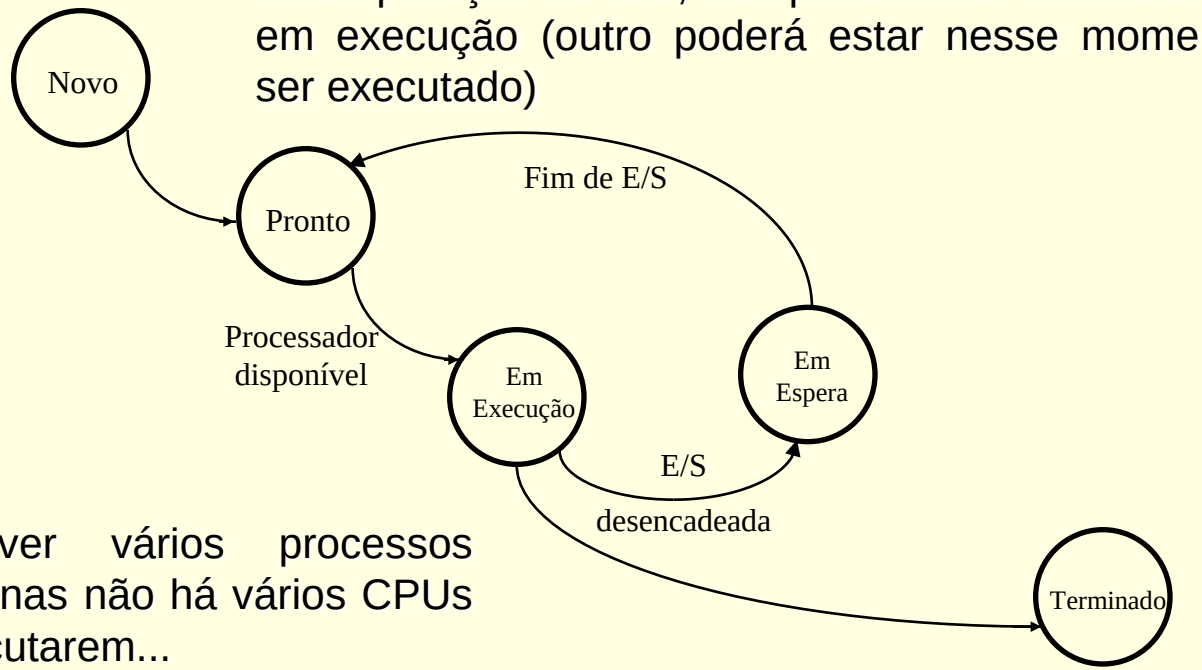
Comutação de Processos desencadeada por I/O

- Vantagens:
  - **Máxima ocupação do CPU:** Um processo corre "produti-vamente" até precisar de I/O.
  - Bom para sistemas *batch*
- Desvantagens:
  - ***Run-away process***: um processo que entra em *loop* sem fazer I/O monopoliza o CPU.
  - Mau para sistemas interactivos

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)

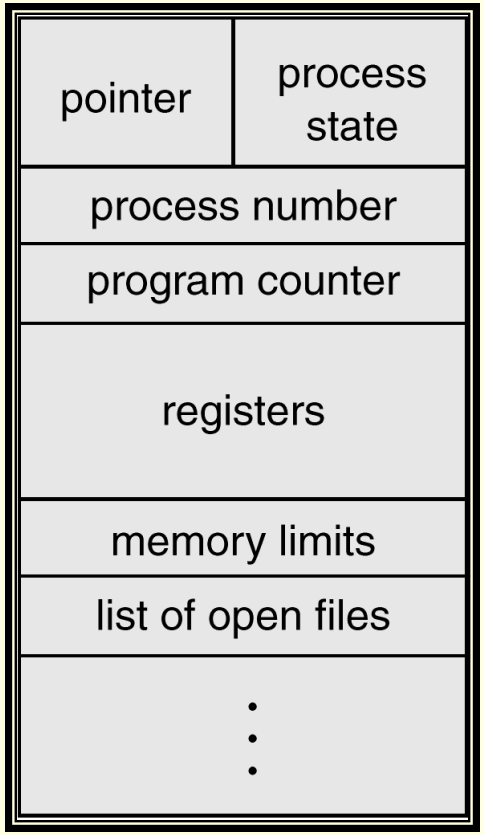


Poderá haver vários processos prontos; apenas não há vários CPUs para os executarem...

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 posteriormente se poder retomar a sua execução.
- Conteúdo do PCB:
  - PC: para retomar a sequência de instruções
  - SP: para recuperar o *stack*
  - Registo de estado: para recuperar as *flags*
  - Registos genéricos / Acumuladores
  - Informação para a GM: RB/RL ou PTBR/PTLR
  - 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



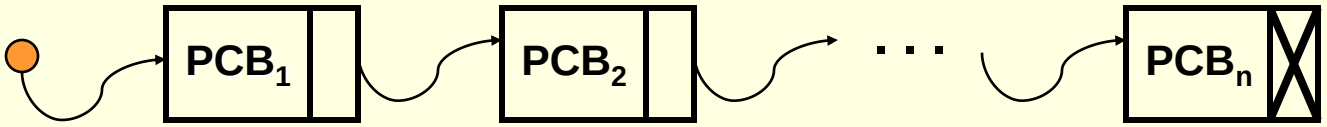
O Process Control Block... mais detalhado

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

Process management	**Memory management**	**File management**
Registers	Pointer to text segment	Root directory
Program counter	Pointer to data segment	Working directory
Program status word	Pointer to stack segment	File descriptors
Stack pointer		User ID
Process state		Group ID
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		

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]

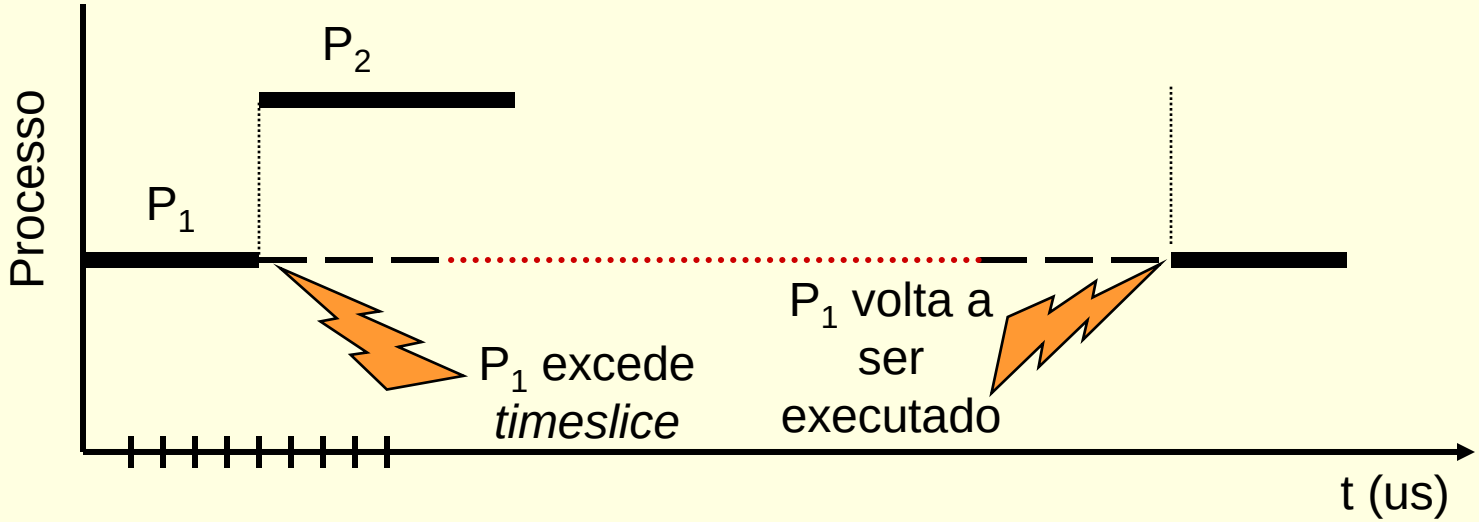


- O escalonador de médio prazo (MTS, *medium term scheduler*) implementa uma politica de ordenação e, portanto, mantém os processos ordenados de acordo com essa política; à cabeça da fila está o "próximo a ser executado". Move os "processos" da fila *waiting* para a *ready*, e intervém em situações de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

Fila dos Processos à Espera

- Fila *de Espera*:
  - Um PCB vai para a "pool" de espera quando o "seu processo" está à espera de um evento – que pode ser multivariado: fim de uma operação de I/O, fim de uma temporização, espera por um recurso,…
  - O MTS move o PCB para a fila *ready* quando termina a situação de espera.

Comutação de Processos por preempção



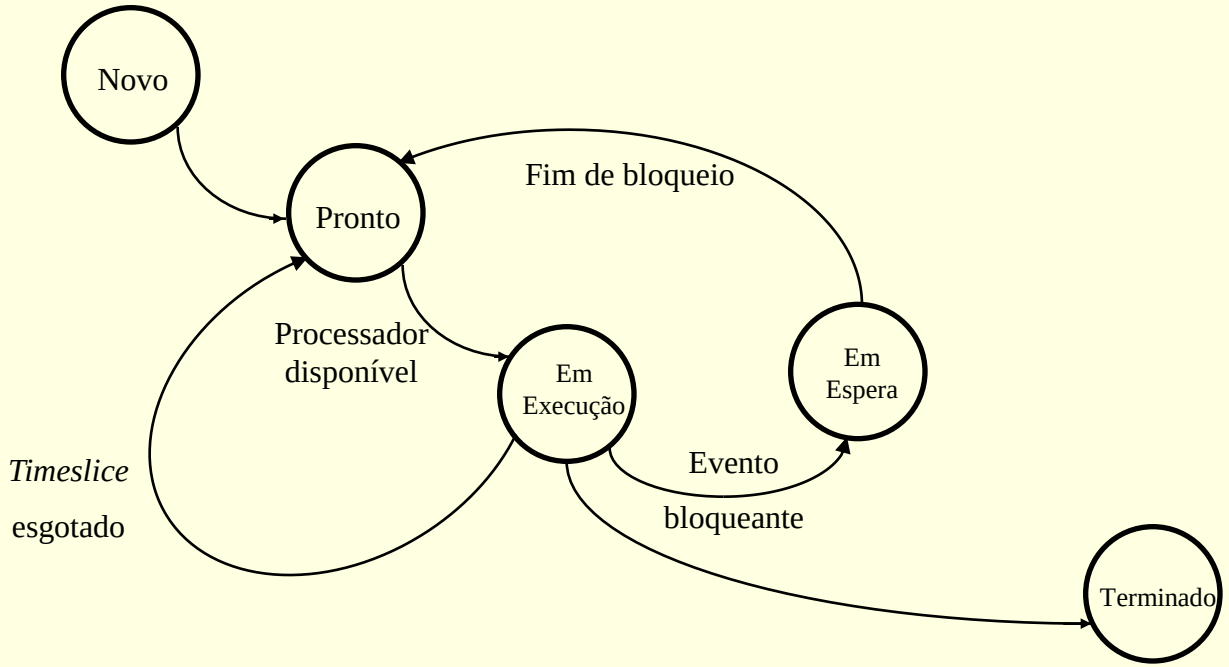
- Quando um processo P_1 excede a sua fatia de tempo (*timeslice*) de execução, há um *interrupt*, o SO é chamado e escolhe um (talvez outro) processo, P_2, para executar.

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 Pág. 20)
  - 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 da “sofisticação” do SO).

Comutação de Processos

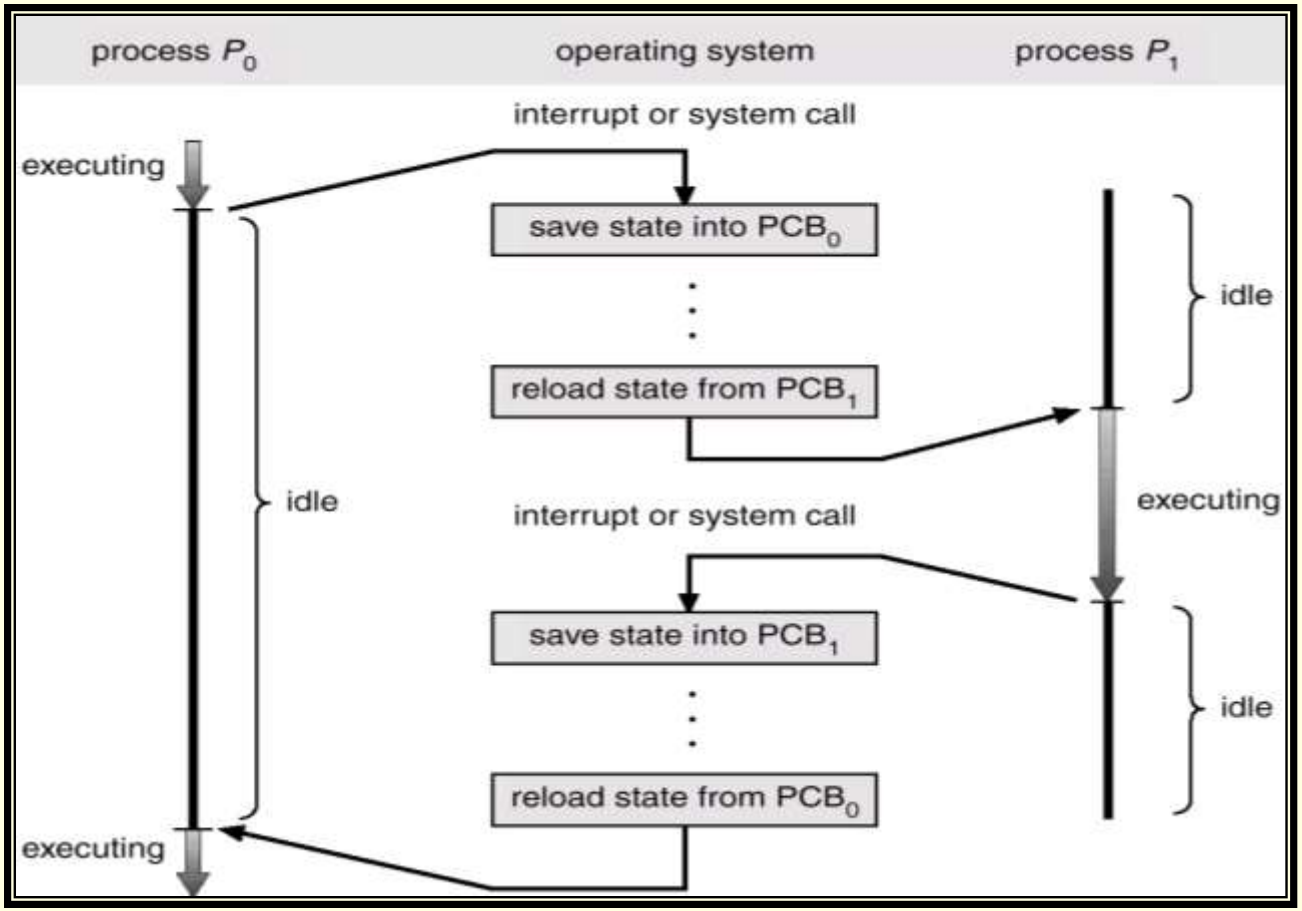
- Diagrama de estados e transições completo para SO's que comutam processos por I/O, ou outro evento bloqueante, e por preempção:



Comutação de Processos

- Desvantagens:
  - A comutação de processos leva tempo, não é instantânea; o tempo gasto na comutação não é "produtivo" (é *overhead*) …
  - Minimizar as comutações (*timeslices* grandes) maximiza o *throughput* … 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: I



Comutação de Processos: II

- A Comutação de Processos (*process switching*, por alguns autores designada *troca de contexto*):
  - Não corresponde a uma actividade útil (é um *overhead*).
  - O tempo gasto para comutar de um processo para outro deve ser minimizado, e é designado latência do dispatcher.
  - O número de comutações deve ser minimizado…

Fundamentos de Sistemas de Operação

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

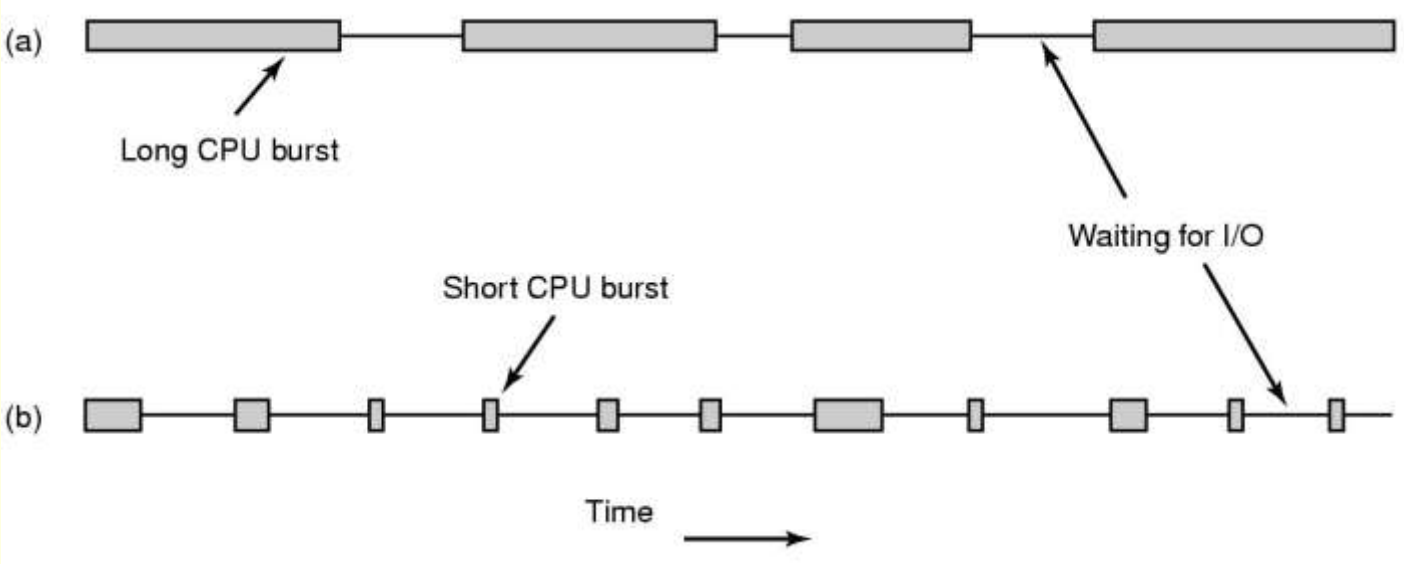
Parte II: Gestão de Processador

b) Escalonamento

Gestão de CPU: factos

- O máximo rendimento do CPU é obtido através da multiprogramação
- A execução dos processos passa sucessivamente por períodos CPU Burst seguidos de I/O Burst
- As características dos processos são muito diferentes e o seu comportamento varia ao longo da sua vida

Gestão de CPU: factos & figuras…



- A execução dos processos passa sucessivamente por períodos CPU Burst seguidos de I/O Burst:

  a) processo CPU-bound
  b) Processo I/O-bound

Gestão de CPU: Critérios de escalonamento

- **Utilização do CPU**: manter o CPU o mais ocupado possível
- ***Throughput*** (débito): maximizar o nº de processos que completam a sua execução por unidade de tempo
- ***Turnaround time*** (Tempo de rotação): minimizar tempo para executar um dado processo
- **Tempo de espera**: minimizar o tempo que um processo esteve à espera para ser executado
- **Tempo de resposta**: minimizar o tempo entre a submissão de pedido e o início da resposta (ambientes interactivos, ou time-sharing)

Gestão de CPU: Critérios de optimização

- **Máxima Utilização do CPU**
- **Máximo *Throughput***
- **Mínimo *Turnaround time***
- **Mínimo Tempo de espera**
- **Mínimo Tempo de resposta**
- **Impossível satisfazer todos simultaneamente!**
  - **Para um dado ambiente de utilização, deve satisfazer-se os que são importantes para esse ambiente...**

Gestão de CPU: Escalonamento

- Objectivo dos Algoritmos de escalonamento:

All systems
- Fairness - giving each process a fair share of the CPU
- Policy enforcement - seeing that stated policy is carried out
- Balance - keeping all parts of the system busy

Batch systems
- Throughput - maximize jobs per hour
- Turnaround time - minimize time between submission and termination
- CPU utilization - keep the CPU busy all the time

Interactive systems
- Response time - respond to requests quickly
- Proportionality - meet users' expectations

Real-time systems
- Meeting deadlines - avoid losing data
- Predictability - avoid quality degradation in multimedia systems