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

c) Escalonamento em Sistemas Batch

Escalonamento em sistemas Batch - I

- SJF (Shortest Job First)

a) Pedidos admitidos (A anterior a B, anterior a C,…)
b) Efeito da reordenação SJF na fila ready

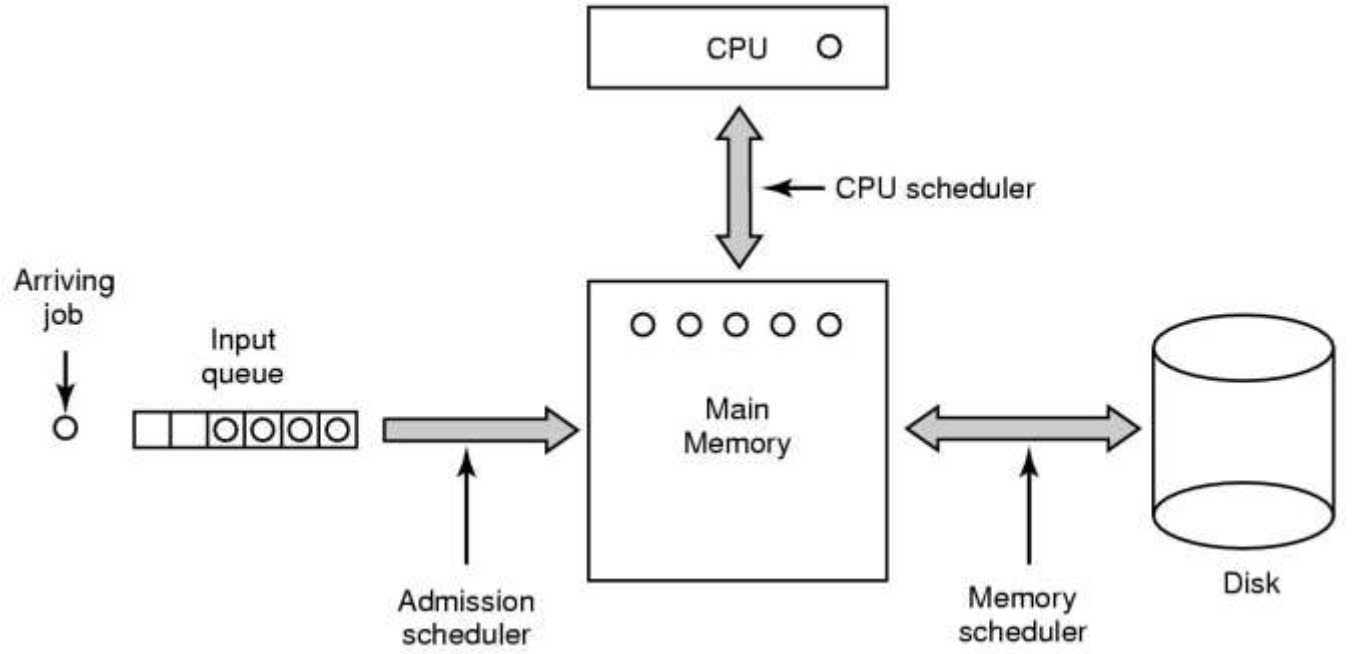
- Como é que se sabe a “duração” de cada *job*?

Escalonamento em sistemas Batch - II

- Políticas de Escalonamento usadas
  - FIFO: favorece ordem de chegada, *throughput*
  - SJF: favorece *turnaround*, *throughput*

Escalonamento em sistemas Batch - III

- Três níveis de escalonamento: longo, médio e curto prazo



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

d) Escalonamento em Sistemas Timesharing

Escalonamento em sistemas TS - I

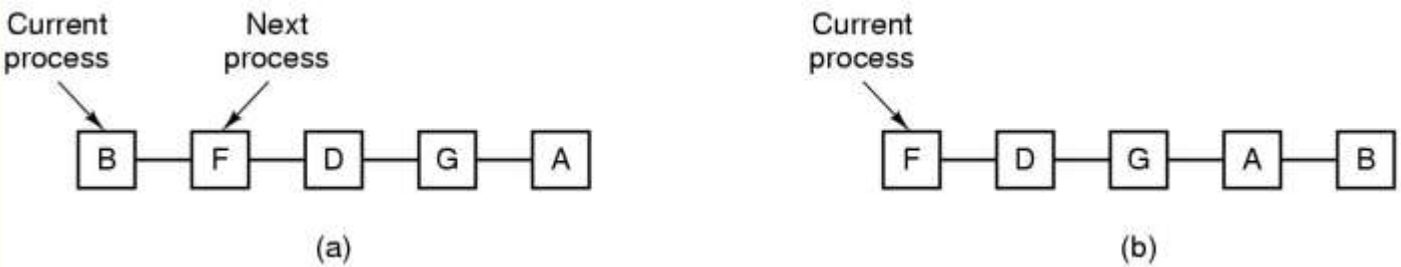
- Objectivos:
  - Justiça (cada processo recebe uma fracção justa do CPU)
  - Equilíbrio (todas as partes do sistema são mantidas ocupadas)
  - Tempo de resposta (minimizado)
  - Proporcionalidade: impedir que um processo (de um utilizador) monopolize o CPU
- Algoritmos de escalonamento baseados em:
  - Fatia de tempo (*timeslice*)
  - Prioridades

Escalonamento em sistemas TS - II

- Escalonamento Round-Robin com *timeslice*
  - Cada processo recebe uma pequena unidade de tempo de CPU (*time quantum* ou *timeslice* – fatia de tempo), usualmente 10 a 100 ms. Após o fim deste tempo, o processo é removido do CPU (*preempted*) e posto na cauda da fila READY.
  - Se há n processos na fila READY e a fatia de tempo é q, cada processo recebe 1/n do tempo de CPU em blocos de q unidades de tempo. Nenhum processo espera mais de (n-1)q unidades de tempo.
  - Se q muito grande, o escalonamento é FIFO; q não pode ser muito pequeno (da ordem de grandeza da latência do dispatcher– tempo que este leva a efectuar a comutação de processos) senão a rentabilidade é muito baixa!

Escalonamento em sistemas TS - II

- Escalonamento rotativo (Round-Robin)



a) lista de processos prontos, B em execução…

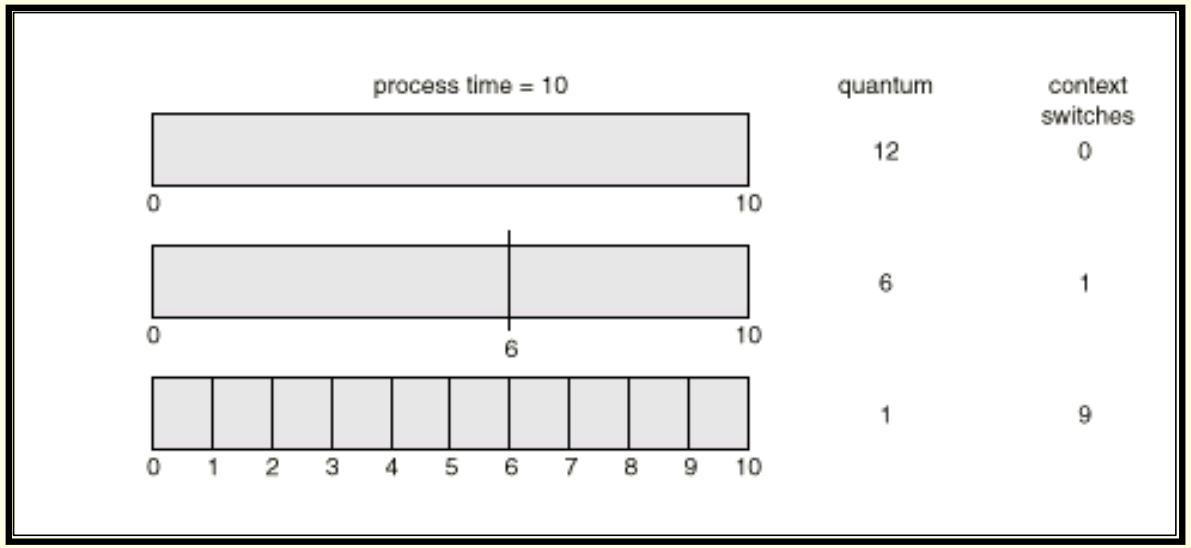
b) lista de processos prontos a correr depois de B usar o seu “quantum” de tempo

Escalonamento em sistemas TS - III

- Escalonamento em sistemas RR com *timeslice*
  - Onde inserir os processos que não esgotam a sua fatia de tempo (porque bloquearam - por terem pedido I/O, ou…)?
    - Depois de serem desbloqueados vão para a cabeça da fila READY, e dependendo da "sofisticação" do SO:
      - recebem um quantum completo, ou
      - recebem o resto do "quantum"
    - favorecimento dos processos I/O *bound* ajuda a manter todas as partes do sistema "equilibradamente" ocupadas

Escalonamento em sistemas TS - III

- Tamanho do *quantum*:
  - q grande ⇒ FIFO
  - q pequeno ⇒ q deve ser muito maior que o tempo de comutação, senão o tempo desperdiçado (*overhead*) é demasiado grande.



Escalonamento baseado em prioridades - I

- *Um valor de prioridade (um número inteiro) está associado a cada processo; o valor é:*
  - *estático, imposto pelo programador e/ou utilizador*
  - *dinâmico: imposto pelo sistema*
  - *misto: estático, mas o sistema pode alterá-lo*
    - *Aumento temporário de prioridade para evitar starvation de um processo*
    - *Variação de prioridade dentro de certos limites – escalonamento Unix*
  - *O CPU é atribuído ao processo com a prioridade mais elevada*
    - *com ou sem preempção*
- *Comum em todos os SOs actuais*
- *Obrigatório em SOs de tempo real*

Escalonamento baseado em prioridades - II

- Problema de *Starvation* – processos com prioridade baixa podem nunca ser executados.
- Solução: *Aging* (envelhecimento) – à medida que um processo vai permanecendo na fila de processos prontos sem lhe ser atribuído o CPU) a sua prioridade vai sendo aumentada; quando por fim lhe é atribuído CPU, a prioridade volta a diminuir.

Escalonamento em sistemas de tempo real

- *Hard real-time systems* – é necessário garantir que uma dada tarefa é completada em menos que um dado limite de tempo *(deadline).*
  - Tal pode ser garantido se conseguirmos ter um limite superior ao tempo máximo que um processo prioritário que necessita de CPU tem de esperar para ser escalonado. O escalonamento é **determinístico**.
  - Exemplos: controlo de tráfego aéreo, de sistemas de energia (estações da rede eléctrica), máquinas ferramentas,...
- *Soft real-time computing* – os processos associados ao processamento de eventos "críticos" têm prioridade sobre os processos "normais".

Outras Políticas de Escalonamento

- Múltiplas filas *Ready*.
  - A fila *Ready* é subdividida em várias filas, com políticas distintas; por exº:
    - *Foreground*: processos interactivos, algoritmo RR.
    - *Background*: processos *batch*, algoritmo FCFS.
  - Escalonamento entre filas:
    - escalonamento com prioridade fixa (i.e., processos *background* só correm se não houver processos *foreground* a precisar de CPU). Possibilidade de *starvation*.
    - Distribuição de tempo – cada fila recebe uma percentagem fixa de tempo de CPU - por ex. 80% para *foreground* (RR) e 20% para *background* (FCFS) .

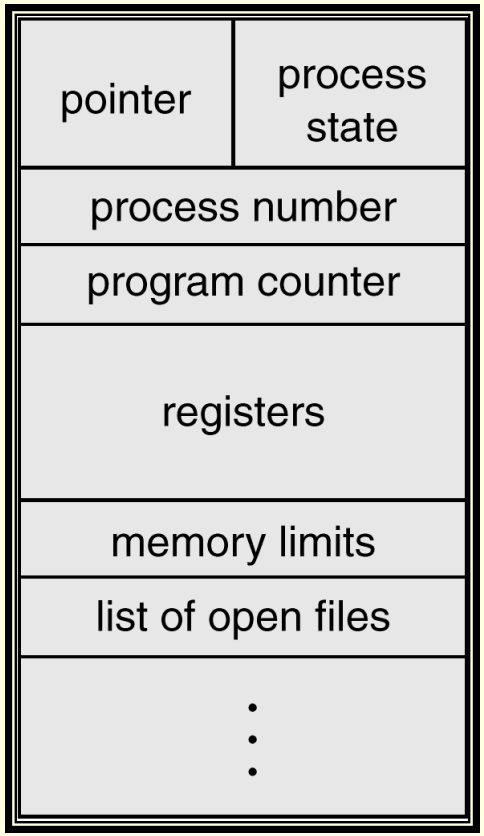
Outras Políticas de Escalonamento

- Escalonamento no Unix System V:
  - Tem por base um escalonamento com *timeslices* de dimensão variável (formadas por um certo nº de *sub-slices*) e prioridades:
    - Processos interactivos, são os que frequentemente estão à espera de I/O (*e.g.*, um editor de texto lê frequentemente do teclado), e raras vezes usam todo o CPU a que têm direito.
    - Processos *batch*, pelo contrário, frequentemente esgotam a fatia de tempo a que têm direito, por precisarem de muito CPU.
  - Algoritmo de Escalonamento heurístico:
    - Favorece os processos interactivos, aumentando a sua prioridade, mas concedendo-lhes *timeslices* pequenas.
    - Diminui a prioridade dos processos batch, deixando-os “mais para trás” na fila *ready*, mas concedendo-lhes *timeslices* maiores – quando por fim chega a sua vez, correm mais tempo sem serem preempted.

Comutação de Processos: PCB (revisão)

- *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 *(revisão)*



O Process Control Block… *(revisão)*

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		

Para pensar…

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		

No escalonamento de threads do mesmo processo, quais são os campos do PCB que têm de ser salvaguardados/carregados?