Gestão de processos

Licenciatura em Tecnologias e Sistemas de Informação

Sistemas Operativos

Helena Rodrigues

(helena@dsi.uminho.pt)

Gestão de processos

Escalonamento

Bibliografia:

Silberschatz, A., Galvin, P., Gagne, Greg, *Applied Operating Systems Concepts*, Addison Wesley, 2000. (1ª edição: cap.6 pp. 135-153; 7ª edição: cap 5 p. 153-169, p. 181-186)

José Alves Marques, Paulo Ferreira, Carlos Ribeiro, Luís Veiga e Rodrigo Rodrigues, Sistemas Operativos, FCA, 2009, cap. 4 p. 121-135

Gestão de processos
Escalonamento

Resultados de aprendizagem:

- Explicar a função e objectivos do gestor de processos no escalonamento de processos, particularmente a função do *dispatcher*.
- Explicar porque é que as diferentes características dos processos são importantes na selecção do algoritmo de escalonamento de processos.
- Descrever as diferentes estratégias de escalonamento.
- Explicar a estratégia de desafectação forçada e as suas consequências para a execução dos programas.
- Descrever os objectivos de um algoritmo de escalonamento e explicar as suas vantagens e desvantagens face às características dos processos

Gestão de processos
Escalonamento

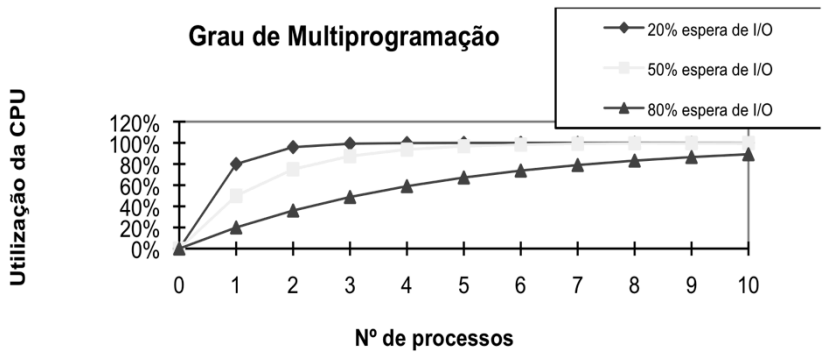
- Num sistema de computação …
  - de um modo geral, um processo/thread representa qualquer actividade em execução, tenha sido criada pelo utilizador ou o próprio SO.
  - estas actividades competem pelo CPU e deste modo surge a necessidade de fazer o seu escalonamento
  - o escalonamento de processos/threads refere-se à tarefa de fazer a gestão da partilha do CPU entre um conjunto de processos/threads prontos a executar

Gestão de processos

Escalonamento

- Em sistemas com um único processador …
  - apenas um processo/thread pode estar em execução num dado momento …
  - …. mas aparentemente executam ao mesmo tempo, por isso dizem-se concorrentes
  - os restantes processos têm de esperar a sua vez mesmo que estejam prontos para serem executados

Multiprogramação (revisão)



Gestão de processos
Escalonamento

- Necessário optimizar utilização do CPU
  - Se um processo se encontra em execução no processador e chega a um ponto do programa onde necessita de executar uma instrução de I/O de dados, isso significa que o CPU não estaria a fazer nada de útil até essa instrução estar terminada
  - Como operações I/O são tipicamente muito mais lentas do que a execução de instruções no CPU, qualquer paragem à espera de dados significa uma quebra substancial no desempenho do sistema
  - CPU deve estar sempre ocupado a executar algum processo e não pode ficar parado à espera que um processo termine I/O
- Se um processo executa uma instrução que o impede, ainda que por breves momentos, de continuar a avançar na execução do seu programa, então tem de ser logo retirado do CPU e substituído por outro que esteja pronto a avançar.

Gestão de processos
Escalonamento

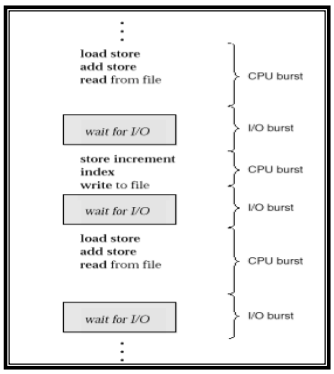
- Objectivos (eficiência)
  - Maximizar a utilização do CPU e outros recursos (100% do tempo ocupado!)
  - Maximizar o número de processos executados por unidade de tempo (***throughput***)
  - Minimizar o tempo de total de execução (***turnaround***)
- Objectivos (conveniência)
  - Justiça
  - Minimizar o **tempo de resposta** aos utilizadores interactivos

Gestão de processos
Escalonamento

- Características dos processos
  - Um processo/thread *pode ser:*
    - Tipo *I/O-bound* ou *CPU-bound*
    - Interactivo ou *batch*
    - Tempo real
    - .....
- Os processos/threads ....
  - podem ter diferentes prioridades num sistema
  - São também caracterizados pelo seu comportamento no sistema (tempo de CPU acumulado, etc)

Gestão de processos
Escalonamento

- A execução de um processo alterna entre:
  - CPU-burst - Períodos de avanço na execução do programa
  - I/O-burst - Períodos (curtos) de bloqueio à espera de sinais, operações de entrada-saída de dados (I/O), etc…

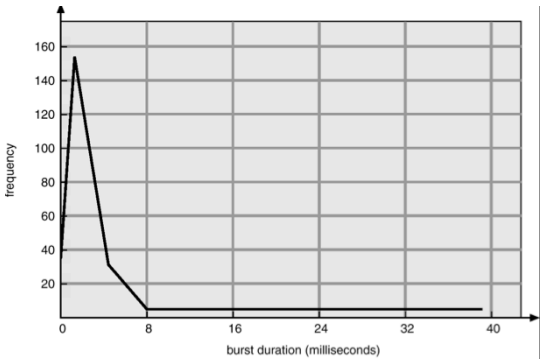


Gestão de processos
Escalonamento

- Duração e números dos CPU *bursts* varia muito de programa para programa
  - CPU-*bound*
    - Processo caracterizado por CPU-bursts mais longos
      - poucas interrupções causada por I/O
  - I/O *bound*
    - Processo caracterizado por muitos CPU-bursts de curta duração
      - Interrupções causadas por I/O são muitas

Gestão de processos
Escalonamento

Histograma típico de CPU-burts

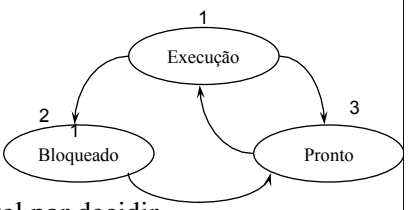


Gestão de processos
Escalonamento

- Dificuldades:
  - As características dos processos podem ser determinantes na selecção do algoritmo de escalonamento mais adequado
  - Os processos têm comportamento imprevisível e natureza heterogénea

Gestão de processos
Escalonamento

- Escalonamento de processos/threads diz respeito ao conjunto de decisões relacionadas:
  - com a transição 3 → 1
  - e, eventualmente com a transição 1 → 3 (*desafectação forçada*)



- Escalonamento de processos é responsável por decidir…
  - **Qual** o próximo processo/thread a executar
  - **Quando** começa esse processo/thread a executar
  - **Durante** quanto tempo esse processo executa

Gestão de processos
Escalonamento

- Estratégias de escalonamento:
  - ***Nonpreemptive scheduling*** ou "*run to completion*";
    - Permite que um processo, uma vez escolhido, execute até que este liberte voluntariamente o CPU (transição 1 → 2) ou termine a execução.
    - Vantagens: Fácil implementação.
    - Problemas: não se adequa a sistemas multi-tarefa (por exemplo sistemas multi-utilizador com *time-sharing*).

Gestão de processos
Escalonamento

- Estratégias de escalonamento:
  - ***Preemptive scheduling*** (desafectação forçada)
    - suspensão de processos, mesmo que tenham todas as condições para executar
    - implica mecanismos de interrupção dos processos.
    - Problemas: as *race conditions* e consequentemente a necessidade de implementar mecanismos complexos como os semáforos e as mensagens...
    - Garantem normalmente tempos de resposta rápidos a processos/threads prioritários ou garantem a partilha equilibrada do CPU pelos diferentes *processos/threads.*

Gestão de processos
Escalonamento

- Níveis de escalonamento:
  - *Long-term scheduler* – selecciona quais os processos a carregar em memória.
    - invocado pouco frequentemente (ordem dos segundos ou minutos) ⇒ moderado na rapidez de execução.
    - controla o grau de multiprogramação
    - normalmente não existe em sistemas de time-sharing
  - *Short-term scheduler* (*CPU scheduler*) – selecciona qual o processo a executar de seguida.
    - invocado frequentemente (na ordem dos milisegundos) ⇒ rapidez na execução.

Gestão de processos
Escalonamento

- *CPU scheduler (Dispatcher)*
  - Escalonamento de curto prazo
  - Objectivo genérico é promover uma utilização produtiva do CPU e do sistema
  - Objectivos específicos determinam políticas de escalonamento diversas
  - selecciona um processo de entre os processos em memória prontos a executar e atribui-lhe o processador.

Gestão de processos
Escalonamento

- Questão
  - Entre o momento em que inicia a sua execução e o momento em que a termina, um processo vai passando esse tempo num destes três estados: bloqueado, execução ou pronto. Quais dos seguintes tempos depende da política de escalonamento utilizada no sistema?
    - Tempo total desde inicio até final (turnaround time)
    - Tempo no estado de execução
    - Tempo no estado de bloqueado
    - Tempo no estado de pronto

Gestão de processos
Escalonamento

- Existem diversos algoritmos de escalonamento adequados a cenários diferentes e que optimizam diferentes variáveis de funcionamento do sistema.
  - *First-Come, First-Served*
  - Round-robin
  - Shortest job first
  - Prioridades
  - Multilevel queue
  - Multilevel feedback queue

Gestão de processos
Escalonamento

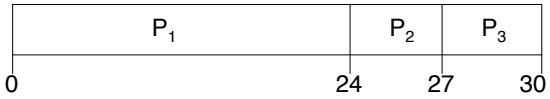
- "*First-Come, First-Served*" (FCFS)
  - O processador é atribuído pela ordem de pedidos dos processos.
  - É implementado por uma fila FIFO
  - O tempo médio de espera de cada processo é, no entanto, muitas vezes longo
  - Por definição é *nonpreemptive*
  - É particularmente inadequado a sistemas de partilha de tempo

Gestão de processos
Escalonamento

- "*First-Come, First-Served*" (FCFS)
  - Exemplo:

Process	Burst Time
P_1	24
P_2	3
P_3	3

  - Submetidos pela seguinte ordem: P_1, P_2, P_3
    O diagrama de Gantt para o escalonamento é:



  - Tempo de espera para P_1 = 0; P_2 = 24; P_3 = 27
  - Média: (0 + 24 + 27)/3 = 17
  - E se a ordem de chegada fosse: P_2, P_3, P_1

Gestão de processos
Escalonamento

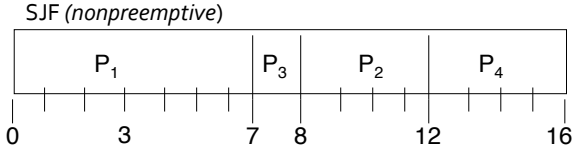
- "*Shortest Job First*" (SJF)
  - Associa a cada processo o tempo de execução do próximo CPU *burst*. Utiliza estes tempos para escalonar o processo com o tempo mais pequeno.
  - Dois esquemas:
    - *nonpreemptive* – assim que o CPU é atribuído a um processo, o processo tem garantia que termina o seu CPU *burst*.
    - *Preemptive* – se um processo novo é submetido com um tempo de CPU *burst* menor que o tempo que falta executar ao processo corrente, *preempt*. Este esquema é conhecido como o *Shortest-Remaining-Time-First (SRTF)*.
  - O SJF é óptimo – produz uma média de tempo de espera mínima para um dado conjunto de processos.

Gestão de processos
Escalonamento

- "*Shortest Job First*" (SJF)
  - Exemplo:

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

  - SJF (*nonpreemptive*)



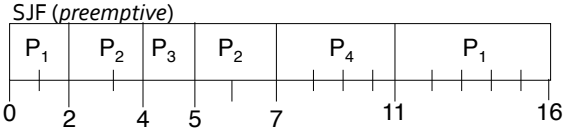
  - Tempo de espera para $P_1 = 0$; $P_2 = 6$; $P_3 = 3$; $P_4 = 7$
  - Média = $(0 + 6 + 3 + 7)/4 = 4$

Gestão de processos
Escalonamento

- "*Shortest Job First*" (SJF)
  - Exemplo:

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

  - SJF (*preemptive*)



  - Tempo de espera para P_1 = 9; P_2 = 1; P_3 = 0; P_4 = 2
  - Média = (9 + 1 + 0 +2)/4 = 3

Gestão de processos
Escalonamento

- Dificuldades do "Shortest Job First" (SJF)
  - Como determinar a duração do próximo CPU *burst*?
  - Nos sistemas *batch* pode ser utilizado o limite de tempo de execução especificado quando um *job* é submetido.
  - Nos sistemas de partilha de tempo podemos aproximar o SJF se utilizarmos uma estimativa do tempo de execução do próximo CPU *burst* calculada com base na duração dos CPU *bursts* anteriores.
    - Deste modo, o processo com o tempo de execução do próximo CPU *burst* previsto menor é o escolhido

Gestão de processos
Escalonamento

- *Scheduling* com prioridades
  - A cada processo é associada uma prioridade (número inteiro); o processador é atribuído ao processo com maior prioridade (normalmente inteiro menor = maior prioridade);
    - *preemptive* - se a prioridade de um processo é mais alta que a prioridade do processo corrente, *preempt*.
    - *Nonpreemptive* - ... o processo de maior prioridade é simplesmente colocado à cabeça da lista de processos prontos.
  - Problema = *Starvation* - processos de baixa prioridade podem não executar nunca
  - Solução ≡ *Aging* – à medida que o tempo progride, a prioridade dos processos que estão à espera de executar incrementa.

Gestão de processos
Escalonamento

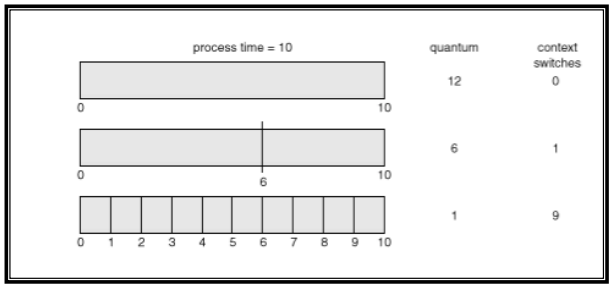
- *Scheduling* com prioridades
  - Atribuição de prioridades:
    - externa - definidas segundo critérios externos ao SO, normalmente associadas a factores económicos e políticos
    - interna - o SO utiliza factores quantificáveis para determinar a prioridade de um processo, como por exemplo, os limites de tempo, os requisitos de memória, o número de ficheiros abertos, a relação entre tempo médio de I/O *burst* e tempo médio de CPU *burst*.
      - Por exemplo dar maior prioridade a processos que utilizam I/O mais frequentemente (executam pouco e esperam muito);

Gestão de processos
Escalonamento

- *Round Robin (RR)*
  - É atribuído a cada processo uma pequeno intervalo de tempo de CPU (*time quantum)*, normalmente 10-100ms. Após este intervalo, o processo é suspenso e inserido no final da fila de processos prontos.
  - Se existem *n* processos na fila de espera e o *quantum* é *q*, então cada processo obtém 1/*n* do tempo do CPU, dividido em quantidades de tempo *q* de cada vez. Nenhum processo espera mais de (*n*-1)*q* unidades de tempo até nova execução.
  - Desempenho
    - *q* grande ⇒ FCFS
    - *q* pequeno ⇒ *q* tem que ser grande relativamente ao tempo de mudança de contexto (overhead)

Gestão de processos
Escalonamento

- *Round Robin (RR)* (cont.)
  - Um *quantum* pequeno incrementa o número de mudanças de contexto.
  - Tempo de execução dos processos varia com o *quantum*:

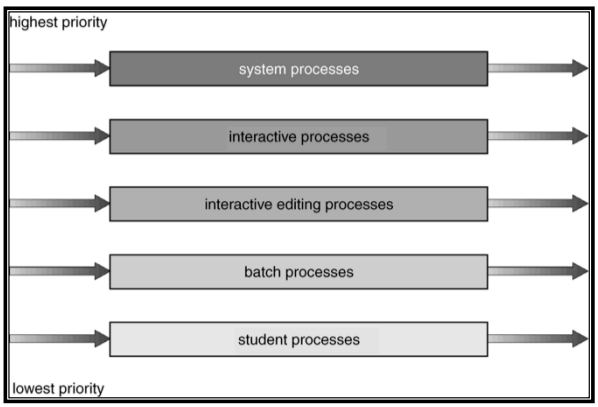


[sil99] - cap. 6

Gestão de processos
Escalonamento

- *Multilevel Queue Scheduling*
  - A fila de processos prontos é dividida em filas separadas. Exemplo: fila dos processos *fg* e fila dos processo *bg*.
  - Cada fila tem o seu algoritmo de *scheduling* próprio. Exemplo: *fg* - RR; *bg* FCFS .
  - É necessário fazer *scheduling* entre filas:
    - prioridade fixa; i.e., apenas executa processos em *bg* se a fila dos processos *fg* está vazia. Situação de *starvation* é possível.
    - Fatia de tempo – é atribuída a cada fila uma fatia de tempo de CPU a qual será dividia pelos processos da fila; i.e., por ex,
      - 80% to fg com RR e 20% to bg com FCFS

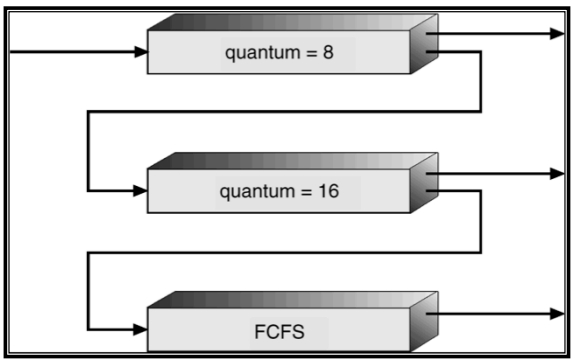
Gestão de processos
Escalonamento



Gestão de processos
Escalonamento

- *Multilevel Feedback Queue*
  - A fila de processos prontos é dividida em filas separadas.
  - Processos podem passar de umas filas para as outras
  - Se um processo usa muito CPU passa para uma fila com menor prioridade
  - Processos interactivos e com muito I/O mantêm-se na fila prioritária.
  - Processos nas filas com menor prioridade vão sendo subidos através de um processo de *aging*

Gestão de processos
Escalonamento



Gestão de processos
Escalonamento

[silboo] - cap. 20, 21 e 22

- Exemplos
  - Unix
    - *round-robin* + prioridades
    - favorece os processos interactivos
    - muita ocupação do CPU implica um decréscimo de prioridade (prioridades dinâmicas)
  - Windows NT
    - "Multiple Queues" com prioridades + *round robin*
    - prioridades dinâmicas

Gestão de processos
Escalonamento

- Linux
  - 2 classes de processos ⇒ 2 algoritmos:
    - algoritmo "*time-sharing*"
    - algoritmo "*real-time*"
  - a identificação dos processos inclui a classe a que pertencem
  - algoritmo "*time-sharing*"
    - baseado na atribuição de créditos e prioridades
    - o crédito de um processo em execução é decrementado em cada *clock interrupt*
    - um processo é executado enquanto tem um crédito > 0
    - quando não existem processos "prontos" com um crédito positivo, o *scheduler* é responsável por atribuir novamente créditos a todos os processos segundo a fórmula:

$$\text{créditos} = \frac{\text{créditos}}{2} + \text{prioridade}$$