


LPRM
Laboratório de Pesquisa em Redes e Multimídia



Processos

(Aula 6)

Escalonamento de Processos

Universidade Federal do Espírito Santo
Departamento de Informática

LPRM Laboratório de Pesquisa em Redes e Multimídia

Aula Passada

- O SO gerencia os recursos do sistema de computação em benefício dos processos
- Para isso, mantém estruturas de controles
 - Tabelas (memória, I/O, arquivos e processos)
- Imagem do processo
 - Programa executável, pilha (stack) e dados
 - BCP (Bloco de Controle do Processo)

Profª. Patrícia D. Costa, LPRM/DI/UFES 2 Sistemas Operacionais 2008/1

LPRM Laboratório de Pesquisa em Redes e Multimídia

Aula Passada (2)

- Há necessidade de intermediar a disputa entre processos no SO
 - Técnicas para gerenciar o processador
 - Técnicas de escalonamento
- Escalonador
 - Longo prazo
 - Curto prazo
 - Médio prazo
- Troca de Contexto

Profª. Patrícia D. Costa, LPRM/DI/UFES 3 Sistemas Operacionais 2008/1

LPRM Laboratório de Pesquisa em Redes e Multimídia

Ciclos de CPU e de I/O ⁽¹⁾

- A execução de um processo consiste numa seqüência intercalada de ciclos de CPU ("CPU Burst cycles") com longos períodos de espera por I/O.
- Esses ciclos de espera por I/O podem ser aproveitados para disparar a execução de outros processos (idéia central da multiprogramação).

Profª. Patrícia D. Costa, LPRM/DI/UFES 4 Sistemas Operacionais 2008/1

Lprm Laboratório de Pesquisa em Redes e Multimídia UFES

Ciclos de CPU e de I/O (2)

The diagram illustrates a sequence of operations grouped into bursts. It starts with a vertical ellipsis, followed by a CPU burst containing 'load store', 'add store', and 'read from file'. This is followed by an I/O burst represented by a box labeled 'wait for I/O'. The sequence then repeats with another CPU burst ('store increment', 'index', 'write to file'), an I/O burst ('wait for I/O'), a third CPU burst ('load store', 'add store', 'read from file'), and a final I/O burst ('wait for I/O'). The diagram ends with another vertical ellipsis. Brackets on the right side group these operations into 'CPU burst' and 'I/O burst' categories.

Prof.ª Patrícia D. Costa, LPRM/DI/UFES 5 Sistemas Operacionais 2008/1

Lprm Laboratório de Pesquisa em Redes e Multimídia UFES

Frequência de Ciclos de CPU (3)

The graph plots frequency on the y-axis (ranging from 0 to 160) against burst duration in milliseconds on the x-axis (ranging from 0 to 40). The curve shows a sharp initial rise to a peak frequency of about 150 at a very short burst duration, followed by a rapid decline. By 8 milliseconds, the frequency has dropped to approximately 10 and remains relatively constant at that low level for the rest of the 40-millisecond duration.

Prof.ª Patrícia D. Costa, LPRM/DI/UFES 6 Sistemas Operacionais 2008/1

Lprm Laboratório de Pesquisa em Redes e Multimídia UFES

Tipos de Escalonadores (Resumo)

Long-term scheduling	The decision to add to the pool of processes to be executed
Medium-term scheduling	The decision to add to the number of processes that are partially or fully in main memory
Short-term scheduling	The decision as to which available process will be executed by the processor
I/O scheduling	The decision as to which process's pending I/O request shall be handled by an available I/O device

Prof.ª Patrícia D. Costa, LPRM/DI/UFES 7 Sistemas Operacionais 2008/1

Lprm Laboratório de Pesquisa em Redes e Multimídia UFES

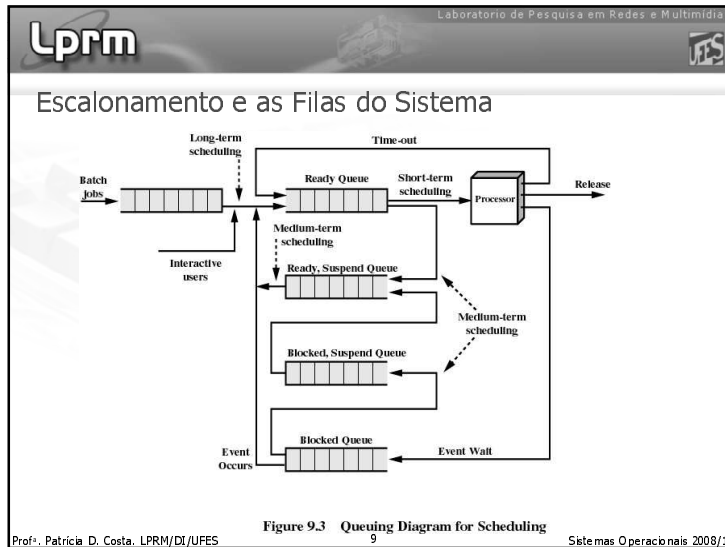
Escalonamento e a Transição de Estados

The diagram shows the following state transitions:

- New** transitions to **Ready/Suspend** and **Ready** via **Long-term scheduling**.
- Ready/Suspend** transitions to **Ready** via **Medium-term scheduling**.
- Ready** transitions to **Running** via **Short-term scheduling**.
- Running** transitions to **EXIT**.
- Blocked/Suspend** transitions to **Blocked** via **Medium-term scheduling**.

Figure 9.1 Scheduling and Process State Transitions

Prof.ª Patrícia D. Costa, LPRM/DI/UFES 8 Sistemas Operacionais 2008/1



- Lprm Laboratório de Pesquisa em Redes e Multimídia
- ## Tipos de Processos
- **Processo CPU Bound:**
 - Uso intensivo de CPU.
 - Realiza pouca operação de E/S.
 - Pode monopolizar a CPU, dependendo do algoritmo de escalonamento.
 - **Processo I/O Bound:**
 - Orientado a I/O.
 - Devolve deliberadamente o controle da CPU.
- Prof.ª Patrícia D. Costa, LPRM/DI/UFES 10 Sistemas Operacionais 2008/1

- Lprm Laboratório de Pesquisa em Redes e Multimídia
- ## Objetivos do Escalonamento
- **Maximizar a taxa de utilização da UCP.**
 - Maximizar a vazão ("*throughput*") do sistema.
 - Minimizar o **tempo de execução** ("*turnaround*").
 - *Turnaround*: tempo total para executar um processo.
 - Minimizar o **tempo de espera** ("*waiting time*"):
 - *Waiting time*: tempo de espera na fila de prontos.
 - Minimizar o **tempo de resposta** ("*response time*").
 - *Response time*: tempo entre requisição e resposta.
- Prof.ª Patrícia D. Costa, LPRM/DI/UFES 11 Sistemas Operacionais 2008/1

- Lprm Laboratório de Pesquisa em Redes e Multimídia
- ## Algoritmos de Escalonamento
- Vários algoritmos podem ser empregados na definição de uma estratégia de escalonamento.
 - Os algoritmos buscam:
 - Obter bons tempos médios ao invés de maximizar ou minimizar um determinado critério.
 - Privilegiar a variância (minimizar) em relação a tempos médios.
 - As políticas de escalonamento podem ser:
 - Preemptivas;
 - Não-preemptivas.
- Prof.ª Patrícia D. Costa, LPRM/DI/UFES 12 Sistemas Operacionais 2008/1

Políticas de Escalonamento

- Preemptivas:
 - O processo de posse da UCP pode perdê-la a qualquer momento, na ocorrência de certos eventos, como fim de fatia de tempo, processo mais prioritário torna-se pronto para execução, etc.
 - Não permite a monopolização da UCP.
- Não-Preemptivas:
 - O processo em execução só perde a posse da UCP caso termine ou a devolva deliberadamente, isto é, uma vez no estado *running*, ele só muda de estado caso conclua a sua execução ou bloqueie a si mesmo emitindo, p.ex., uma operação de E/S.

Exemplos de Algoritmos

- FIFO (First-In First-Out) ou FCFS (First-Come First-Served).
- SJF (Shortest Job First) ou SPN (Shortest Process Next).
- Round-Robin.
- Priority.
- Multiple queue.

First-Come First-Served ⁽¹⁾

- Algoritmo de baixa complexidade.
- Exemplo de abordagem não-preemptiva.
- Processos que se tornam aptos para execução são inseridos no final da fila de prontos.
- O primeiro processo da fila é selecionado para execução.
- O processo executa até que:
 - Termina a sua execução;
 - Realiza uma chamada ao sistema.

First-Come First-Served ⁽²⁾

- Processos pequenos podem ter que esperar por muito tempo, atrás de processos longos, até que possam ser executados ("*convoy effect*").
- Favorece processos CPU-bound.
 - Processos I/O-bound têm que esperar até que processos CPU-bound terminem a sua execução.
- Algoritmo particularmente problemático para sistemas de tempo compartilhado, onde os usuários precisam da CPU a intervalos regulares.

First-Come First-Served ⁽³⁾

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suponha que os esses processos cheguem na seguinte ordem:
 - P_1, P_2, P_3

First-Come First-Served ⁽⁴⁾

- Tempo de espera para cada processo:
 - *Waiting time:* $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
- Tempo médio de espera:
 - *Average waiting time:* $(0 + 24 + 27)/3 = 17$

First-Come First-Served ⁽⁵⁾

- Suponha que os mesmos processos cheguem agora na seguinte ordem:
 - P_2, P_3, P_1
- Tempo de espera de cada processo:
 - *Waiting time:* $P_1 = 6$; $P_2 = 0$; $P_3 = 3$
- Tempo médio de espera:
 - *Average waiting time:* $(6 + 0 + 3)/3 = 3$

Shortest Job First ⁽¹⁾

- Baseia-se no fato de que privilegiando processos pequenos o tempo médio de espera decresce.
 - O tempo de espera dos processos pequenos decresce mais do que o aumento do tempo de espera dos processos longos.
- É um algoritmo ótimo, de referência.

Shortest Job First ⁽²⁾

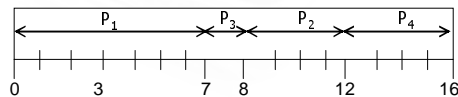
- Associado com cada processo está o tamanho do seu próximo *CPU burst*.
- Esse tamanho é usado como critério de escalonamento, sendo selecionado o processo de menor próximo *CPU burst*.

Shortest Job First ⁽³⁾

- Dois esquemas:
 - Não-preemptivo – uma vez a CPU alocada a um processo ela não pode ser dada a um outro antes do término do CPU burst corrente.
 - Preemptivo – se chega um novo processo com CPU burst menor que o tempo remanescente do processo corrente ocorre a preempção. Esse esquema é conhecido como Shortest-Remaining-Time-First (SRTF).

Exemplo de SJF Não-Preemptivo

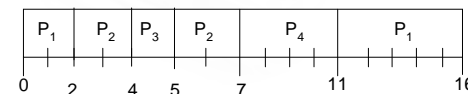
Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



- *Average waiting time* = $(0 + 6 + 3 + 7)/4 = 4$

Exemplo de SJF Preemptivo (Algoritmo SRTF)

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



- *Average waiting time* = $(9 + 1 + 0 + 2)/4 = 3$

Tamanho do Próximo *CPU burst*

- A real dificuldade do algoritmo é conhecer o tamanho da próxima requisição de CPU.
 - Para escalonamento de longo prazo num sistema batch, podemos usar como tamanho o limite de tempo de CPU especificado pelo usuário quando da submissão do job.
 - No nível de escalonamento de curto prazo sua implementação pode ser apenas aproximada, já que não há como saber o tamanho da próxima requisição de CPU.

Prevedendo o Tamanho do *Burst* ⁽¹⁾

- Uma maneira de se aproximar do SJF é *prevedendo* o tamanho do próximo CPU burst.
- Normalmente isso é feito usando uma *média exponencial* das medidas dos *bursts* anteriores.

1. t_n = actual length of n^{th} CPU burst
2. τ_{n+1} = predicted value for the next CPU burst
3. $\alpha, 0 \leq \alpha \leq 1$
4. Define:

$$\tau_{n+1} = \alpha t_n + (1 - \alpha)\tau_n$$

Prevedendo o Tamanho do *Burst* ⁽²⁾

- O parâmetro α controla o peso relativo do histórico recente e passado na fórmula.
 - Se $\alpha = 0$
 - $\tau_{n+1} = \tau_n$
 - Histórico recente não é considerado relevante.
 - Se $\alpha = 1$
 - $\tau_{n+1} = t_n$
 - Apenas o último CPU burst é levado em conta.

Prevedendo o Tamanho do *Burst* ⁽³⁾

- Se a fórmula for expandida, teremos:

$$\begin{aligned} \tau_{n+1} = & \alpha t_n + (1 - \alpha) \alpha t_{n-1} + \dots \\ & + (1 - \alpha)^j \alpha t_{n-j} + \dots \\ & + (1 - \alpha)^{n+1} \tau_0 \end{aligned}$$

- Como tanto α quanto $(1 - \alpha)$ são menores ou iguais a 1, cada termo sucessivo possui menos peso que o seu predecessor.

Lprm Laboratório de Pesquisa em Redes e Multimídia

Prevedendo o Tamanho do *Burst* (4)

$\alpha = 1/2$
 $\tau_0 = 10$

CPU burst (t_i)	6	4	6	4	13	13	13	...
"guess" (τ_t)	10	8	6	6	5	9	11	12

Prof.ª Patrícia D. Costa, LPRM/DI/UFES 29 Sistemas Operacionais 2008/1

Lprm Laboratório de Pesquisa em Redes e Multimídia

Escalonamento por Prioridade (1)

- Um número inteiro é associado a cada processo, refletindo a sua prioridade no sistema.
- A CPU é alocada ao processo de maior valor de prioridade na fila de prontos.
 - OBS: normalmente, menor valor = maior prioridade
- Estratégia muito usada em S.O. de tempo real.
- Problema: "*starvation*"
 - Processos de baixa prioridade podem nunca executar
- Solução: "*Aging*"
 - Prioridade aumenta com o passar do tempo.

Prof.ª Patrícia D. Costa, LPRM/DI/UFES 30 Sistemas Operacionais 2008/1

Lprm Laboratório de Pesquisa em Redes e Multimídia

Escalonamento por Prioridade (2)

- Prioridades podem ser definidas interna ou externamente.
 - Definição interna:
 - Usa alguma medida (ou uma combinação delas) para computar o valor da prioridade. Por exemplo, limite de tempo, requisitos de memória, n° de arquivos abertos, razão entre *average I/O burst* e *average CPU burst*, etc.
 - Definição externa:
 - Definida por algum critério externo ao S.O (tipo do processo, departamento responsável, custo, etc.)

Prof.ª Patrícia D. Costa, LPRM/DI/UFES 31 Sistemas Operacionais 2008/1

Lprm Laboratório de Pesquisa em Redes e Multimídia

Escalonamento por Prioridade (3)

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

P2	P5	P1	P3	P4
0	1	6	16	18

Average waiting time = $(6+0+16+18+1)/5 = 8,2ms$

Prof.ª Patrícia D. Costa, LPRM/DI/UFES 32 Sistemas Operacionais 2008/1

Lprm Laboratório de Pesquisa em Redes e Multimídia

Escalonamento por Prioridade (3)

Process	Burst Time	Priority
P_1 background	10	1
P_2 Interativo	1	0
P_3 Interativo	2	0
P_4 Interativo	1	0
P_5 background	5	1

Average waiting time = $(4+0+1+3+14)/5 = 4,4\text{ms}$

Prof.ª Patrícia D. Costa, LPRM/DI/UFES 33 Sistemas Operacionais 2008/1

Lprm Laboratório de Pesquisa em Redes e Multimídia

Escalonamento *Round-Robin* (1)

- Algoritmo típico de sistemas operacionais de tempo compartilhado.
- Cada processo recebe uma pequena fatia de tempo de CPU (quantum), usualmente entre 10 e 100 ms.
- Após o término da sua fatia de tempo o processo é "interrompido" e colocado no final da fila de prontos ("preempção" baseada na interrupção de relógio).
- É um algoritmo justo???

 - Dependendo do tamanho do quantum, processos CPU bound tendem a ganhar mais tempo de processador que processos I/O bound

Prof.ª Patrícia D. Costa, LPRM/DI/UFES 34 Sistemas Operacionais 2008/1

Lprm Laboratório de Pesquisa em Redes e Multimídia

Escalonamento *Round-Robin* (2)

- Se n processos existem na fila de prontos e a fatia de tempo é q , então cada processo recebe $1/n$ do tempo de CPU, em fatias de q unidades de tempo de cada vez.
- Nenhum processo espera mais do que $(n-1) \cdot q$ unidades de tempo.
- Potencialmente pode ter um tempo de resposta melhor
 - Se q for um pouco maior que o tempo requerido para uma interação típica (ou uma função), o tempo de resposta médio será próximo de q
 - Caso contrário, mínimo de $2q$ seria necessário
- Tipicamente, apresenta um tempo de *turnaround* médio maior que o SJF, por que? Troca de contexto

Prof.ª Patrícia D. Costa, LPRM/DI/UFES 35 Sistemas Operacionais 2008/1

Lprm Laboratório de Pesquisa em Redes e Multimídia

Desempenho do Algoritmo

- Dependente do tamanho do quantum:
 - q grande \Rightarrow tende a FIFO.
 - q pequeno \Rightarrow gera muito *overhead* devido às trocas de contexto.

Prof.ª Patrícia D. Costa, LPRM/DI/UFES 36 Sistemas Operacionais 2008/1

Lprm Laboratório de Pesquisa em Redes e Multimídia UFES

Turnaround x Tamanho do Quantum

- O tempo de *turnaround* varia com o tamanho do quantum.
- O *turnaround* médio de um conjunto de processos não necessariamente diminui quando o tamanho do quantum é aumentado (vira FCFS).
- Regra geral: 80% CPU burst < q

process	time
P_1	6
P_2	3
P_3	1
P_4	7

Prof.ª Patrícia D. Costa, LPRM/DI/UFES 37 Sistemas Operacionais 2008/1

Lprm Laboratório de Pesquisa em Redes e Multimídia UFES

Escalonamento Multinível (1)

- A idéia base é dividir os processos em diferentes grupos, com diferentes requisitos de tempos de resposta.
- A cada grupo é associada uma fila de prioridade, conforme a sua importância .

Prof.ª Patrícia D. Costa, LPRM/DI/UFES 38 Sistemas Operacionais 2008/1

Lprm Laboratório de Pesquisa em Redes e Multimídia UFES

Escalonamento Multinível (2)

- Por exemplo, a fila de prontos pode dividida em duas filas separadas:
 - *foreground* (p/ processos interativos)
 - *background* (p/ processamento *batch*)
- Cada fila apresenta o seu próprio algoritmo de escalonamento:
 - *foreground* – RR
 - *background* – FCFS

Prof.ª Patrícia D. Costa, LPRM/DI/UFES 39 Sistemas Operacionais 2008/1

Lprm Laboratório de Pesquisa em Redes e Multimídia UFES

Escalonamento Multinível (3)

- Escalonamento deve ser feito entre as filas:
 - Prioridades fixas – atende primeiro aos processos da fila *foreground* e somente depois aos da fila *background*.
 - Time slice – cada fila recebe uma quantidade de tempo de CPU para escalonamento entre os seus processos. Ex: 80% para *foreground* em RR e 20% para *background* em FCFS.

Prof.ª Patrícia D. Costa, LPRM/DI/UFES 40 Sistemas Operacionais 2008/1

Escalonamento Multinível com Feedback

- O processo pode se mover entre as várias filas. Deste modo, a estratégia de *aging* pode ser implementada.
- O escalonador trabalha com base nos seguintes parâmetros:
 - Número de filas;
 - Algoritmo de escalonamento de cada fila;
 - Método usado para determinar quando aumentar e quando reduzir a prioridade do processo;
 - Método usado para se determinar em que fila o processo será inserido.

Exemplo (1)

- Suponha a existência de 3 filas:
 - Q_0 – time quantum 8 milliseconds
 - Q_1 – time quantum 16 milliseconds
 - Q_2 – FCFS
- Escalonamento:
 - Um job novo entra na fila Q_0 , que é servida segundo a estratégia RR. Quando ele ganha a CPU ele recebe 8 ms. Se não terminar em 8 ms, o job é movido para a fila Q_1 .
 - Em Q_1 o job é novamente servido RR e recebe 16 ms adicionais. Se ainda não completar, ele é interrompido e movido para a fila Q_2 .
 - Em Q_2 , FCFS

Exemplo (2)

