Victor Augusto Costa Monteiro
NºUSP 8942937
victor.augusto.monteiro@usp.br

Lista de Exercícios 2

Sistemas Operacionais - ACH

22 de novembro de 2023

1. Diferencie os escalonamentos preemptivos e não-preemptivos.

Um algoritmo de escalonamento não preemptivo escolhe um processo para executar e vai deixá-lo executando até que ele termine, bloqueie ou que voluntariamente libere a CPU, ou seja, ele não vai suspender o processo em execução.

Um algoritmo preemptivo pode suspender o processo em execução. Ele tem algum tipo de controle de uso da CPU, seja por tempo, por ciclo de clock ou qualquer outro tipo de controle. Ao acabar seu tempo de uso, ele bloqueia o processo e escolhe um outro processo para usar a CPU.

2. Os computadores CDC 6600 podiam lidar simultaneamente com até 10 processos de E/S, usando uma forma interessante de round robin chamada compartilhamento de processador. Um chaveamento de processo ocorria depois de cada instrução; assim, a instrução 1 vinha do processo 1, a instrução 2 do processo 2 etc. O chaveamento era feito por um hardware especial e a sobrecarga era zero. Se um processo precisasse de T segundos para terminar sua execução, na ausência de competição, quanto tempo seria necessário se o compartilhamento do processador fosse usado com n processos?

O escalonamento por round robin usando o quantum como a execução de uma instrução, a concorrência com n processos faz um processo de tempo T segundos executar em $n * T$ segundos.

3. Escalonadores round-robin normalmente mantém uma lista de todos os processos que podem ser rodados, com cada processo ocorrendo exatamente uma vez na lista. O que aconteceria se um processo ocorresse duas vezes na lista? Você consegue pensar em uma razão para que se permita isso?

Um processo que esteja duas vezes na lista de round robin executaria duas vezes ao longo de um round, ou seja, usaria a CPU duas vezes mais que os outros processos na lista. Um motivo para isso é uma prioridade maior para esse processo.

4. As medidas de um certo sistema mostram que o processo médio executa por um tempo T antes de ser bloqueado para E/S. Um chaveamento de processos requer um tempo S efetivamente gasto (sobrecarga). Para o escalonamento circular (round robin) com um quantum Q, dê uma fórmula para a eficiência (ou seja, o tempo útil dividido pelo tempo total de CPU) da CPU, para rodar um processo médio, em cada um dos seguintes casos:

No geral, a fração dada para cálculo da eficiência da CPU é $\frac{Q}{Q+S}$, o tempo de uso da CPU pelo processo (o quantum) dividido pela soma do uso da CPU pelo processo e pelo uso do round robin para a troca de contexto.

1. $Q = \infty$

   Maximiza a eficiência da CPU para que os processos só liberem quando bloquearem ou terminarem, a troca de contexto ocorre o mínimo de vezes, apenas quando é estritamente necessário. Mas os processos que entrarem por último vão demorar muito tempo para serem executados. E elimina o caráter preemptivo do algoritmo round robin, vira quase que um first in first out. Seria um algoritmo pouco responsivo. Eficiência dada por $\frac{T}{T+S}$.

2. $Q > T$

   É o mesmo do item acima, vai minimizar a troca de contexto e otimizar o uso da CPU para o processo parar só quando acabar ou bloquear por E/S, mas o caráter preemptivo do round robin é reduzido .Eficiência dada por $\frac{T}{T+S}$.

3. $S < Q < T$

   Otimiza o uso da CPU sem comprometer o caráter preemptivo do algoritmos, ele fica mais responsivo para novas entradas de processos na fila. Não vai minimizar as trocas de processos, mas no caso de não

ter uma operação E/S para o bloqueio, é garantido que o processo vai usar a CPU por mais tempo que a troca de contexto. Ficará entre $\frac{1}{2}$, que é o caso $S = Q$, e $\frac{Q}{Q+S}$, que é o caso $Q = T$.

4. $Q = S$

   Uso de 50%, $\frac{Q}{Q+S} = \frac{Q}{Q+Q} = \frac{Q}{2Q} = \frac{1}{2}$

5. Q próximo de 0

   Minimiza a eficiência da CPU, a troca de contexto vai ocupar a maior parte do tempo de uso da CPU. Eficiência vai tender a 0, a CPU será a usada a maior parte do tempo para a troca de contexto da execução dos processos.

5. Em um sistema batch, cinco tarefas estão esperando para serem executadas. Seus tempos de execução previstos são 9, 6, 3, 5 e X. Em que ordem elas deveriam ser executadas para minimizar o tempo médio de resposta?

O melhor turnaround possível, sabendo os tempos de antemão, é com o algoritmo do Shortest Job First, na ordem, 3, 5, 6, 9 e o X por último por não saber o tempo de execução desse processo.

6. Cinco tarefas em lote, A a E, chegam a um centro de computação quase ao mesmo tempo. Elas têm tempos de execução estimados em 10, 6, 2, 4 e 8 min. Suas prioridades, determinadas externamente, são 3, 5, 2, 1 e 4, respectivamente, com 5 sendo a mais alta. Para cada um dos seguintes algoritmos, determine o tempo médio de execução completa (mean turnaroundtime) desses processos. Ignore o tempo gasto com a troca de processos.

Dado um quantum de 1 minutos. Como todos chegaram ao mesmo tempo, o tempo inicial de todas é zero.

1. Round Robin

   Pressupõe igual prioridade para todos os processos.

$$T = \frac{t_A+t_B+t_C+t_D+t_E}{Processos} = \frac{(30-0)+(23-0)+(8-0)+(17-0)+(28-0)}{5} = \frac{94}{5} = 21.2minutos$$

2. Escalonamento por prioridades

   Reordena por prioridade 6, 8, 10, 2 e 4 minutos. E roda cada um até terminar e passa pro próximo.

$$T = \frac{t_A+t_B+t_C+t_D+t_E}{Processos} = \frac{(24-0)+(6-0)+(26-0)+(30-0)+(14-0)}{5} = \frac{100}{5} = 20.0minutos$$

3. First-come, First-served (na ordem 10, 6, 2, 4, 8)

   Vai rodar o primeiro até o final, o segundo até o final e aí por diante. Ignora a prioridade.

$$T = \frac{t_A+t_B+t_C+t_D+t_E}{Processos} = \frac{(10-0)+(16-0)+(18-0)+(22-0)+(30-0)}{5} = \frac{96}{5} = 19.2minutos$$

4. Shortest job first

   Reordenar por tempo de execução crescente 2, 4, 6, 8, 10. Ignora a prioridade.

$$T = \frac{t_A+t_B+t_C+t_D+t_E}{Processos} = \frac{(30-0)+(12-0)+(2-0)+(6-0)+(20-0)}{5} = \frac{70}{5} = 14.0minutos$$

7. Um processo rodando com um escalonador de múltiplas filas (CTSS) necessita de 30 quanta para ser finalizado. Quantas vezes ele será colocado para rodar, incluindo a primeira vez (antes que tenha começado a rodar)?

Cada vez que um processo é colocado para rodar ele recebe o dobro de quanta da última vez que rodou. Pra somar 30 quanta ele vai rodar 1, 2, 4, 8 e 15 (de 16 possíveis no último), o que soma 5 vezes que o escalador colocou o processo para rodar.

8. Um sistema de tempo real tem quatro eventos periódicos com períodos de 50, 100, 200 e 250 ms cada. Suponha que os quatro eventos requeiram 35, 20, 10 e x ms de tempo de CPU, respectivamente. Qual o maior valor de x para que o sistema seja escalonável?

O sistema em tempo real escalonável se

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

Onde P_i é o período de tempo em que ocorre o evento e C_i é o tempo para tratar a ocorrência dos eventos.

Então o maior valor de x é tal que $\frac{35}{50} + \frac{20}{100} + \frac{10}{200} + \frac{x}{250} \leq 1$, logo, $x \leq 12.5ms$

9. Com o valor de x do exercício anterior, ilustre o escalonamento dos processos segundo
(a) Rate Monotonic Scheduling
(b) Earliest Deadline First
Para tanto, suponha que todos os processos tentam rodar no instante 0.

Para o Rate Monotonic Scheduling, atribui uma prioridade igual à frequência de ocorrência do evento. Eveto A, com periodicidade 50 ms tem prioridade 20 (e_1). Evento B, com periodicidade 100 ms tem prioridade 10 (e_2). Evento C, com periodicidade 200 ms tem prioridade 5 (e_3). Evento D, com periodicidade 250 ms tem prioridade 4 (e_4).

Processo A pode interromper B, C e D, processo B pode interromper C e D e o processo C pode interromper D. Processo desenvolvido na Figura 1.

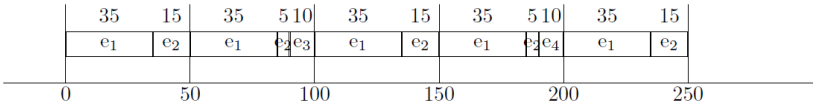

Figura 1: Rate Monotonic Scheduling - Em 250 nova requisição de (e_1) e (e_4) chegam. Na fila estão (e_2) (parcial), (e_3) e (e_4) (parcial). Estourou o tempo para (e_4).

Para o Earliest Deadline First, as prioridades são de acordo com o prazo para entrada do processo novamente na fila. O processo que vai vencer mais cedo é o que tem prioridade para entrar na execução. Mostrado na Figura 2

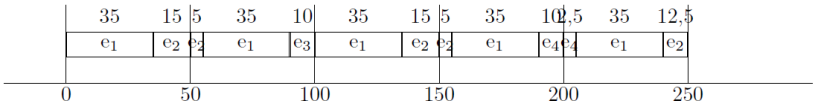

Figura 2: Earliest Deadline First - não estoura o prazo do processo D.

10. Um sistema de tempo real precisa controlar duas chamadas de voz, cada uma executada a cada 5 ms e consumindo 1 ms do tempo da CPU por surto, além de um vídeo de 25 quadros/s, sendo que cada quadro requer 20 ms do tempo da CPU. Esse sistema pode ser escalonado? Justifique.

Um sistema em tempo real é escalonável se

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

$\frac{1}{5} + \frac{1}{5} + \frac{20}{40} \leq 1$, logo, $0.9 \leq 1$

Portanto, é escalonável.

11. Ilustre o escalonamento dos processos do exercício anterior segundo
    (a) Rate Monotonic Scheduling
    (b) Earliest Deadline First
    Para tanto, suponha que todos os processos tentam rodar no instante 0.

Para o Rate Monotonic Scheduling, atribui uma prioridade igual à frequência de ocorrência do evento. Evento 1, com periodicidade 5 ms tem prioridade 200. Evento 2, com periodicidade 5 ms tem prioridade 200. Evento V, com periodicidade 40 ms tem prioridade 50.

Processo desenvolvido na Figura 3.

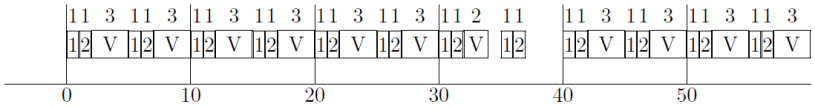

Figura 3: Rate Monotonic Scheduling

Para o Earliest Deadline First, as prioridades são de acordo com o prazo para entrada do processo novamente na fila. O processo que vai vencer mais cedo é o que tem prioridade para entrar na execução. O resultado é igual ao do RMS.

12. Considere que cinco processos sejam criados no instante de tempo 0 (P1, P2, P3, P4 e P5) e possuam as características descritas na tabela 4 a seguir. Qual o tempo de turnaround médio segundo os seguintes algoritmos?

Processo	Tempo de UCP	Prioridade
P_1	10	3
P_2	14	4
P_3	5	1
P_4	7	2
P_5	20	5

Figura 4: Tabela do exercício

- First Come, First Served

  Ignora a prioridade.

$$T = \frac{P_1+P_2+P_3+P_4+P_5}{Processos} = \frac{(10-0)+(24-0)+(29-0)+(36-0)+(56-0)}{5} = \frac{155}{5} = 31ut$$

- Shortest job first

  Reordena os processos em ordem crescente de tratamento.

$$T = \frac{P_3+P_4+P_1+P_2+P_5}{Processos} = \frac{(5-0)+(12-0)+(22-0)+(36-0)+(56-0)}{5} = \frac{131}{5} = 26.2ut$$

- Prioridade (número menor implica prioridade maior)

$$T = \frac{P_3+P_4+P_1+P_2+P_5}{Processos} = \frac{(5-0)+(12-0)+(22-0)+(36-0)+(56-0)}{5} = \frac{131}{5} = 26.2ut$$

- Circular com fatia de tempo igual a 2 u.t. (pressupondo multiprogramação)

$$T = \frac{P_3+P_4+P_1+P_2+P_5}{Processos} = \frac{(25-0)+(34-0)+(38-0)+(48-0)+(56-0)}{5} = \frac{201}{5} = 40.2ut$$

13. Considere um sistema operacional que implemente escalonamento circular (round robin) com fatia de tempo igual a 10 u.t.. Em um determinado instante de tempo, existem apenas três processos (P1, P2 e P3) na fila de prontos, e o tempo de CPU de cada processo é 18, 4 e 13 u.t, respectivamente. Qual o estado de cada processo no instante de tempo T, considerando a execução dos processos P1, P2 e P3, nesta ordem, e que nenhuma operação de E/S é realizada?

- $T = 8ut$

  Primeiro quanta, P_1 rodando, P_2 e P_3 prontos para rodar.

- $T = 11ut$

  Segundo quanta, P_2 rodando, P_1 e P_3 prontos para rodar.

- $T = 33ut$

  Aos 33 ut o P_3 está em execução e os P_1 e P_2 foram concluídos.

14. Considere um sistema operacional que implemente escalonamento circular com fatia de tempo igual a 10 u.t. Em um determinado instante T = 0 u.t., existem apenas três processos (P1, P2 e P3) na fila de pronto, e o tempo de CPU de cada processo é 14, 4 e 12 u.t, respectivamente. Qual o estado de cada processo no instante de tempo T, considerando a execução dos processos P1, P2 e P3, nesta ordem, e que apenas o processo P1 realiza operações de E/S? Cada operação de E/S é executada a cada 5 u.t. e consome 10 u.t.

- $T = 8ut$

  P_1 bloqueado, P_2 rodando e P_3 pronto para rodar.

- $T = 18ut$

  P_3 rodando, P_1 pronto para rodar e P_2 concluído.

- $T = 28ut$

  P_1 bloqueado, P_2 e P_3 foram concluídos.

15. Existem quatro processos (P1, P2, P3 e P4) na fila de pronto, com tempos de CPU estimados em 9, 6, 3 e 5, respectivamente. Em que ordem os processos devem ser executados para minimizar o tempo de turnaround dos processos?

    O mínimo do turnaround ocorre com o Shortest Job First, então P_3, P_4, P_2 e P_1.

16. Considere a tabela 5 a seguir. Qual o tempo de turnaround médio dos processos considerando o tempo de troca de contexto igual a 0 e a 5 u.t. para os seguintes escalonamentos.

Processo	Tempo de UCP	Prioridade
P_1	40	4
P_2	20	2
P_3	50	1
P_4	30	3

Figura 5: Tabela do exercício

- FCFS - First Come, First Served

  Troca com 0 ut

$$T = \frac{P_1+P_2+P_3+P_4}{Processos} = \frac{(40-0)+(60-0)+(110-0)+(140-0)}{4} = \frac{350}{4} = 87.5ut$$

Troca com 5 ut

$$T = \frac{P_1+P_2+P_3+P_4}{Processos} = \frac{(40-0)+(65-0)+(120-0)+(155-0)}{4} = \frac{380}{4} = 95ut$$

- SJF

  Troca com 0 ut

$$T = \frac{P_1+P_2+P_3+P_4}{Processos} = \frac{(90-0)+(20-0)+(140-0)+(50-0)}{4} = \frac{300}{4} = 75ut$$

  Troca com 5 ut

$$T = \frac{P_1+P_2+P_3+P_4}{Processos} = \frac{(100-0)+(20-0)+(155-0)+(55-0)}{4} = \frac{330}{4} = 82.5ut$$

- Circular com fatia de tempo igual à 20 ut

  Troca com 0 ut

$$T = \frac{P_1+P_2+P_3+P_4}{Processos} = \frac{(100-0)+(40-0)+(140-0)+(130-0)}{4} = \frac{410}{4} = 102.5ut$$

  Troca com 5 ut

$$T = \frac{P_1+P_2+P_3+P_4}{Processos} = \frac{(120-0)+(45-0)+(175-0)+(160-0)}{4} = \frac{480}{4} = 125ut$$

17. Seguindo o algoritmo de loteria, mostre a ordem de escalonamento dos processos acima, sabendo que a cada um é atribuído um número de bilhetes igual à sua prioridade, na ordem P1 . . .P4, e que os bilhetes sorteados são 10, 5, 1, 9, 4, 2, 6, 8, 3, 7.

Processo	Prioridade
P_1	4
P_2	2
P_3	1
P_4	3

Figura 6: Tabela do exercício

Considerando o menor número como o de maior prioridade:
P_1: bilhetes 1 a 4
P_2: bilhetes 5 e 6
P_3: bilhetes 7
P_4: bilhetes 8 a 10

Ordem do resultado é P_4, P_2, P_1, P_4, P_1, P_1, P_2, P_4, P_1 e P_3

18. Um usuário possui 5 processos (A_1 a A_5), enquanto que outro possui 2 (B_1 e B_2). Ilustre uma possível ordem de escalonamento, para o algoritmo fair-share, em que são feitas as seguintes promessas ao usuário:

- Cada usuário terá a mesma quantia de CPU

  Metade para cada um vai alternar um processo de cada usuário.

  $A_1, B_1, A_2, B_2, A_3, B_1, A_4, B_2, A_5, B_1$ etc.

- O usuário A terá o dobro do tempo de CPU que o usuário B

  Dois processos de A seguido por um processo de B.

  $A_1, A_2, B_1, A_3, A_4, B_2, A_5$ etc.

- O usuário B terá o triplo do tempo de CPU que o usuário A

  Três processos de B e um processo de A.

  $B_1, B_2, B_1, A_1, B_1, B_2, B_1, A_2, B_1, B_2, B_1, A_3, B_1, B_2, B_1, A_4, B_1, B_2, B_1, A_5$ etc.

19. Em um sistema operacional, o escalonador utiliza duas filas. A fila A contém os processos do pessoal do CPD e a fila B contém os processos dos alunos. O algoritmo dentro das filas é fatia de tempo (Round Robin). De cada 11 unidades de tempo de processador, 7 são fornecidas para os processos da fila A, e 4 para os processos da fila B. O tempo de cada fila é dividido entre os processos também por fatias de tempo, com fatias de 2 unidades para todos. A tabela abaixo mostra o conteúdo das duas filas no instante zero. Considere que está iniciando um ciclo de 11 unidades, e agora a fila A vai receber as suas 7 unidades de tempo. Mostre a sequência de execução dos processos, com os momentos em que é feita a troca.
    Obs. Se terminar a fatia de tempo da fila X no meio da fatia de tempo de um dos processos, o processador passa para a outra fila. Entretanto, esse processo permanece como primeiro da fila X e recebe novo quantum.

	Fila A			Fila B		
Processo	P_1	P_2	P_3	P_4	P_5	P_6
Duração	6	5	7	3	8	4

Figura 7: Tabela do exercício

A			B			*Tempo*	*Obs.*
P_1	P_2	P_3	P_4	P_5	P_6	*Final*	
6	5	7	3	8	4	0	Situação inicial. Começará em A
4	3	5	-	-	-	6	Somente rodam em A
3	-	-	-	-	-	7	Em 7, troca de fila
-	-	-	1	6	-	11	Em 11 troca de fila. Recomeça em P_1 (que tinha sido interrompido)
1	1	3	-	-	-	17	
0	-	-	-	-	-	18	Em 18, P_1 termina, e troca de fila
	-	-	-	-	2	20	
	-	-	0	-	-	21	Em 21 P_4 termina
	-	-		5	-	22	Em 22 troca de fila
	0	-		-	-	23	Em 23 P_2 termina
		1		-	-	25	
		0		-	-	26	Em 26, P_3 termina. Não há mais nada em A
				3	0	30	Em 30, P_6 termina
				1		32	
				0		33	Em 33, P_5 termina

Figura 8: Tabela do resultado