SO - Projeto 2

Cristian de Andrade Custodio,
Leonardo Nunes Guimarães Costa,
Samuel Vitor Pedro e Araujo

[1]Engenharia de Computação e Informação
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

1. Introdução

Neste PDF, estão as perguntas e respostas do Projeto 2, referentes aos capítulos 7 e 8 do livro utilizado no curso de Sistemas Operacionais. Desenvolvemos um script em Python para executar cada aplicação, gerando um arquivo para cada pergunta. Para cada pergunta, você pode encontrar mais detalhes das simulações nos arquivos de saída (”output”).

2. Capítulo 7

1. **Calcule o tempo de resposta e o tempo de retorno ao executar três trabalhos de duração 200 com os escalonadores SJF e FIFO.**

   Para três trabalhos de duração 200:

FIFO:

Job	Tempo de Resposta	Tempo de Retorno	Tempo de Espera
0	0.00	200.00	0.00
1	200.00	400.00	200.00
2	400.00	600.00	400.00

Médias:

Tempo de Resposta	Tempo de Retorno	Tempo de Espera
200.00	400.00	200.00

- **Tempo de Resposta:** 0, 200, 400
- **Tempo de Retorno:** 200, 400, 600

SJF:

Job	Tempo de Resposta	Tempo de Retorno	Tempo de Espera
0	0.00	200.00	0.00
1	200.00	400.00	200.00
2	400.00	600.00	400.00

Médias:

Tempo de Resposta	Tempo de Retorno	Tempo de Espera
200.00	400.00	200.00

- **Tempo de Resposta:** 0, 200, 400
- **Tempo de Retorno:** 200, 400, 600

2. **Faça o mesmo, mas com trabalhos de durações diferentes: 100, 200 e 300.**

   Para trabalhos de durações 100, 200 e 300:

FIFO:

Job	Tempo de Resposta	Tempo de Retorno	Tempo de Espera
0	0.00	100.00	0.00
1	100.00	300.00	100.00
2	300.00	600.00	300.00

Médias:

Tempo de Resposta	Tempo de Retorno	Tempo de Espera
133.33	333.33	133.33

- **Tempo de Resposta:** 0, 100, 300
- **Tempo de Retorno:** 100, 300, 600

SJF:

Job	Tempo de Resposta	Tempo de Retorno	Tempo de Espera
0	0.00	100.00	0.00
1	100.00	300.00	100.00
2	300.00	600.00	300.00

Médias:

Tempo de Resposta	Tempo de Retorno	Tempo de Espera
133.33	333.33	133.33

- **Tempo de Resposta:** 0, 100, 300
- **Tempo de Retorno:** 100, 300, 600

3. **Faça o mesmo, mas também com o escalonador RR e um time-slice de 1.**

   Para trabalhos de durações 200 com RR (Round Robin) e time-slice de 1:

RR:

Job	Tempo de Resposta	Tempo de Retorno	Tempo de Espera
0	0.00	598.00	398.00
1	1.00	599.00	399.00
2	2.00	600.00	400.00

Médias:

Tempo de Resposta	Tempo de Retorno	Tempo de Espera
1.00	599.00	399.00

- **Tempo de Resposta:** 0, 1, 2
- **Tempo de Retorno:** 598, 599, 600

Para trabalhos de durações 100, 200 e 300 com RR (Round Robin) e time-slice de 1:

RR:

Job	Tempo de Resposta	Tempo de Retorno	Tempo de Espera
0	0.00	298.00	198.00
1	1.00	499.00	299.00
2	2.00	600.00	300.00

Médias:

Tempo de Resposta	Tempo de Retorno	Tempo de Espera
1.00	465.67	265.67

- **Tempo de Resposta:** 0, 1, 2
- **Tempo de Retorno:** 298, 499, 600

4. **Para que tipos de cargas de trabalho o SJF entrega os mesmos tempos de retorno que o FIFO?**

   O SJF entrega os mesmos tempos de retorno que o FIFO para trabalhos de mesma duração entre si e (consequentemente) ordem (Ex: 200, 200, 200), ou para trabalhos de duração diferente entre si e com a ordenação do FIFO sendo em ordem crescente de tamanho. Como é possível ver nos exercícios 1 e 2, pois a ordem de execução dos trabalhos será a mesma em ambos os escalonadores.

5. **Para que tipos de cargas de trabalho e comprimentos de quantum o SJF entrega os mesmos tempos de resposta que o RR?**

   Todos os processos onde tem o mesmo tempo de duração que o quantum do RR. Se cada processo leva exatamente o mesmo tempo para rodar e esse tempo é igual ao quantum do RR, ambos os algoritmos finalizam os processos no mesmo instante.

   Vejamos um exemplo de um SJF de duração 200 e um RR de quantum 200:

SJF:

Job	Tempo de Resposta	Tempo de Retorno	Tempo de Espera
0	0.00	200.00	0.00
1	200.00	400.00	200.00
2	400.00	600.00	400.00

Médias:

Tempo de Resposta	Tempo de Retorno	Tempo de Espera
200.00	400.00	200.00

RR:

Job	Tempo de Resposta	Tempo de Retorno	Tempo de Espera
0	0.00	200.00	0.00
1	200.00	400.00	200.00
2	400.00	600.00	400.00

Médias:

Tempo de Resposta	Tempo de Retorno	Tempo de Espera
200.00	400.00	200.00

6. **O que acontece com o tempo de resposta no SJF à medida que a duração dos trabalhos aumenta? Você pode usar o simulador para demonstrar a tendência?**

   À medida que a duração dos trabalhos aumenta, o tempo de resposta no SJF tende a aumentar para trabalhos mais longos, pois o SJF sempre seleciona o trabalho mais curto disponível. Trabalhos mais longos ficam esperando na fila até que todos os trabalhos mais curtos sejam concluídos.

Usando o simulador para demonstrar essa tendência:

SJF 1:

Job	Tempo de Resposta	Tempo de Retorno	Tempo de Espera
0	0.00	1.00	0.00
1	1.00	2.00	1.00
2	2.00	3.00	2.00

Médias:

Tempo de Resposta	Tempo de Retorno	Tempo de Espera
1.00	2.00	1.00

SJF 100:

Job	Tempo de Resposta	Tempo de Retorno	Tempo de Espera
0	0.00	100.00	0.00
1	100.00	200.00	100.00
2	200.00	300.00	200.00

Médias:

Tempo de Resposta	Tempo de Retorno	Tempo de Espera
100.00	200.00	100.00

SJF 1000:

Job	Tempo de Resposta	Tempo de Retorno	Tempo de Espera
0	0.00	1000.00	0.00
1	1000.00	2000.00	1000.00
2	2000.00	3000.00	2000.00

Médias:

Tempo de Resposta	Tempo de Retorno	Tempo de Espera
1000.00	2000.00	1000.00

7. **O que acontece com o tempo de resposta no RR à medida que os comprimentos do quantum aumentam? Você pode escrever uma equação que dê o tempo de resposta no pior caso, dado N trabalhos?**

   À medida que o comprimento do quantum aumenta, o tempo de resposta no RR também aumenta, especialmente para trabalhos que chegam no final da fila. Isso ocorre porque cada trabalho precisa esperar mais tempo até que o quantum retorne a ele. A equação para o tempo de resposta no pior caso para N trabalhos, dado um quantum q, pode ser expressa como:

   $$T_{\text{max}} = (N - 1) \times q$$

   Isso ocorre porque no pior caso, o trabalho precisa esperar que todos os outros N-1 trabalhos sejam atendidos antes de ele ser atendido.

   Podemos usar o exemplo da questão 3, onde temos 3 processos ($N = 3$) e um quantum 1 ($q = 1$):

   $$T_r = (3 - 1) \times 1 = 2 \times 1 = 2$$

   Neste exemplo, o tempo de resposta no pior caso é 2:
   Job 2 – Response: 2.00

3. Capítulo 8

1. **Execute alguns problemas gerados aleatoriamente com apenas dois trabalhos e duas filas; calcule o rastreamento de execução do MLFQ para cada um. Faça sua vida mais fácil limitando a duração dos trabalhos e desligando I/O.**

   Configuração:
   Dois trabalhos (-j 2),
   Duas filas (-n 2),
   Limitando tamanho (-m 2),
   Desativando I/O (-M 0).

   Comando:
   run_mlfq(['-n','2','-j','2','-m','2','-M','0','-l','0,30,0:20,20,0','-c'], outfile)

   Rastreamento de execução:
   - De T = 0 a T = 10 a tarefa 0 executa com prioridade 1.
   - De T = 10 a T = 20 a tarefa 0 executa com prioridade 0.
   - De T = 20 a T = 30, a tarefa 1 chega com prioridade 1.
   - De T = 30 a T = 40 a tarefa 0 executa até o fim com prioridade 0.
   - A partir de T = 40 a tarefa 1 executa até o fim.

   Para mais informações, consulte o arquivo output_cap8_1.txt.

2. **Como você executaria o escalonador para reproduzir cada um dos exemplos do capítulo?**

   Exemplo 1: A Single Long-Running Job

   Configuração:
   3 filas (-n 3),
   Início: 0, Duração: 200, I/O: 0

   Comando:
   run_mlfq(['-n', '3', '-l', '0,200,0', '-c'], outfile)

   Exemplo 2: Along Came A Short Job

   Configuração:
   3 filas (-n 3),
   Tarefa B chega no T=100ms e dura por 20ms.

   Comando:
   run_mlfq(['-n', '3', '-l', '0,180,0:100,20,0', '-c'], outfile)

   Exemplo 3: What About I/O?

   Configuração:
   3 filas (-n 3),
   Ativação da propriedade da regra 4b (-S),
   Tarefa B sendo executada desde T=50ms, 25 vezes, com I/O de 1ms a cada 5ms.

   Comando:
   run_mlfq(['-S', '-n', '3', '-l', '0,175,0:50,25,1', '-i', '5', '-c'], outfile)

3. **Como você configuraria os parâmetros do escalonador para se comportarem exatamente como um escalonador round-robin?**

   Definindo o número de filas no MLFQ como 1. Isso cria uma fila única, eliminando o comportamento multinível do MLFQ e simulando o round-robin, onde todos os processos são tratados igualmente.

   Com isso, temos a atuação da regra 2:
   Rule 2: If Priority(A) = Priority(B), A e B run in RR.

   Comando:
   run_mlfq(['-n', '1', '-q', '5', '-l', '0,10,0:0,10,0', '-c'], outfile)

4. **Crie uma carga de trabalho com dois jobs e parâmetros do escalonador para que um trabalho aproveite as antigas Regras 4a e 4b (ativadas com a flag -S) para manipular o escalonador e obter 99% da CPU durante um determinado intervalo de tempo.**

   Regra 4a: Uma tarefa que usa todo seu quantum, terá sua prioridade reduzida.
   Regra 4b: Uma tarefa que abandonar a CPU (por exemplo, executando uma operação de I/O) antes que seu quantum termine, permanecerá no mesmo nível de prioridade.

   Comando:
   run_mlfq(['-S', '-n', '2', '-l', '0,1000,9:0,1000,0', '-i', '0', '-c'], outfile)

   Tarefa 0: Executa por 1000ms. Realiza I/O a cada 9ms de duração 0, nunca usando todo o quantum de 10ms. Pela Regra 4b, permanece no mesmo nível de prioridade.
   Tarefa 1: Executa por 1000ms. Usa todo o quantum de 10ms. Pela Regra 4a, tem sua prioridade reduzida.

   Como resultado, a Tarefa 0 continua a executar por 9ms, faz I/O, e permanece no mesmo nível de prioridade sem ser interrompida pela Tarefa 1. Isso permite que a Tarefa 0 monopolize a CPU por quase todo o tempo (99%), aproveitando as Regras 4a e 4b, e execute até o ciclo 1010.

5. **Dado um sistema com comprimento quântico de 10 ms em sua fila mais alta, com que frequência você teria que impulsionar os empregos de volta à prioridade mais alta nível (com a flag -B) para garantir que um único trabalho de longa duração (e potencialmente faminto) obtenha pelo menos 5% da CPU?**

   Para garantir que uma tarefa de longa duração obtenha pelo menos 5% da CPU, impulsionamos a tarefa de volta ao nível de prioridade mais alto a cada 200 ms. Em um período de 200 ms, 5% corresponde a 10 ms de execução (5% de 200 ms = 10 ms). Portanto, promovendo a tarefa a cada 200 ms, garantimos que ela obtenha 10 ms de execução nesse intervalo, atendendo ao requisito de 5% de uso da CPU.

   EX:
   run_mlfq(['-n', '2', '-B', '200','-l', '0,200,0:0,200,0', '-c'], outfile)

6. **Uma questão que surge no escalonador é qual final da fila adicionar um trabalho que acabou de concluir a I/O; a flag -I altera esse comportamento para este simulador de escalonador. Experimente algumas cargas de trabalho e veja se você consegue ver o efeito desse sinalizador.**

   Com a flag -I, uma tarefa volta a executar assim que terminar o I/O, caso não tenha terminado seu quantum.

   run_mlfq(['-i','1', '-n', '2','-l', '0,20,5:0,20,5', '-c'], outfile)
   run_mlfq(['-I','-i','1', '-n', '2', '-l', '0,20,5:0,20,5', '-c'], outfile)

   Devido à flag -I no segundo caso. A tarefa que completou uma operação de I/O continuou a ser executado sem ser movido para o final da fila, impactando ligeiramente a distribuição do tempo de CPU entre as tarefas e resultando em tempos de turnaround levemente diferentes em comparação ao caso sem a flag -I.

4. Link para GitHub

Clique abaixo para ir ao repositório do GitHub:

Ir para o repositório GittCris - SO.