Universidade do Estado de Santa Catarina – Bacharelado em Ciência da Computação
Sistemas Operacionais – Prof. Rafael R. Obelheiro – Semestre: 2020.1

Trabalho 4 – Gerência de E/S & Deadlocks

Sobre o trabalho

- O trabalho é individual.
- Exercícios de implementação devem ser realizados na linguagem C, e serão testados no sistema operacional Linux. As submissões estarão sujeitas a controle de plágio usando ferramentas de detecção de similaridade.
- Exercícios teóricos devem ser submetidos em formato PDF. Para os exercícios de implementação deve ser submetido o código fonte, em formato compilável (ou seja, os arquivos `*.c`, e não listagens em PDF). Submeta um único arquivo ZIP contendo todas as suas respostas (teóricas e de implementação).
- O trabalho deverá ser entregue até **SEGUNDA-FEIRA, 21 DE SETEMBRO**. O Moodle aceitará submissões até 23h59min, sendo automaticamente bloqueado após a data limite. É **RESPONSABILIDADE DOS ALUNOS** garantir que o trabalho seja entregue no prazo.
- Em caso de dificuldades ou dúvidas na interpretação do trabalho, entre em contato com o professor (`rafael.obelheiro@udesc.br`).

Exercícios

1. Suponha que um disco tem 2000 cilindros, numerados de 1 a 2000, e um tempo de deslocamento entre trilhas adjacentes de 0,5 ms. O disco está atualmente atendendo a uma requisição no cilindro 210, sendo que a última requisição atendida foi no cilindro 212. A fila de requisições pendentes, em ordem FIFO, é dada pelos números obtidos em `https://bit.ly/2RgJkW9`. Partindo da posição atual do cabeçote de leitura e gravação, determine o tempo total de leitura (em ms) para o conjunto de requisições (excluindo a requisição do cilindro 210) quando o escalonamento do disco é feito usando os algoritmos FCFS, SSF e elevador, sabendo que o disco tem uma velocidade de rotação de 6000 rpm, setores de 512 bytes e 128 setores por trilha, e que em cada requisição são lidos 8 KB de setores consecutivos. Para o algoritmo SSF, caso haja cilindros equidistantes do atual, o braço do disco deve continuar no mesmo sentido do último movimento.
2. Em um sistema existem três processos (A, B e C) e quatro recursos não compartilháveis (W, X, Y e Z). A ordem em que os processos requisitam e liberam a alocação desses recursos é determinada pelos pseudocódigos abaixo.
   (a) Mostre uma sequência de execução que provoque um *deadlock* envolvendo os três processos. Use os números das linhas para representar a sequência (se a linha L1 contém uma requisição, use a notação $L1^+$ para indicar que o recurso é alocado, e $L1^-$ para indicar que o processo não consegue alocar o recurso).
   (b) Desenhe o grafo de alocação de recursos que mostra a ocorrência do *deadlock* caracterizado no item (a).
   (c) Mostre como a possibilidade de ocorrência de *deadlock* pode ser eliminada modificando a ordem das linhas em um ou mais processos. Considere que cada processo precisa usar simultaneamente todos os recursos que solicita (ou seja, não é possível alterar as linhas A4, B3 e C4).

Processo A	
A1	requisita W
A2	requisita X
A3	requisita Z
A4	usa W+X+Z
A5	libera W
A6	libera X
A7	libera Z

Processo B	
B1	requisita Y
B2	requisita Z
B3	usa Y+Z
B4	libera Y
B5	libera Z

Processo C	
C1	requisita W
C2	requisita Z
C3	requisita Y
C4	usa W+Z+Y
C5	libera Y
C6	libera Z
C7	libera W

3. Escreva um programa C que simule parte de um escalonador de disco. O programa deve atender aos seguintes requisitos:

   R1. O programa deve ler requisições de disco da entrada padrão (uma requisição por linha). O formato das requisições é mostrado abaixo, juntamente com alguns exemplos:

bloco inicial	*no. de blocos*	*operação*
101	5	r
75	4	w
106	3	r
99	2	w
109	60	r

   O *bloco inicial* corresponde ao primeiro bloco a ser lido ou escrito, e o *no. de blocos* contém a quantidade de blocos (limitada a 64 por requisição); ambos são valores inteiros sem sinal. A *operação* pode ser `r`, para leitura, e `w`, para escrita. Nos exemplos, a primeira requisição solicita a leitura dos blocos 101 a 105, e a segunda a escrita dos blocos 75 a 78.

   R2. A leitura encerra quando o bloco inicial for −1. Essa requisição deve ser ignorada.

   R3. As requisições lidas devem ser inseridas em uma fila de despacho, que deve ser ordenada pelo número do bloco.

   R4. Requisições com a mesma operação que sejam adjacentes devem ser fundidas em uma única requisição, respeitando o limite para o número de 64 blocos por requisição. Também devem ser fundidas requisições de leitura onde haja sobreposição de blocos.

   R5. As requisições devem ser fundidas à medida em que forem sendo lidas (i.e., não apenas ao final da leitura).

   R6. Não há limite predefinido para o tamanho da lista de requisições.

   R7. Quando a leitura for encerrada, o programa deve imprimir uma linha contendo "Fila:" e exibir o conteúdo da lista, uma requisição por linha, com campos separados por um espaço (veja o exemplo abaixo).

   Exemplo de execução do programa:

```
$ ./req-disco
100 2 r
75 4 w
106 5 r
9 4 w
99 2 w
111 60 r
101 5 r
6 3 w
10 2 w
-1
Fila:
6 7 w
10 2 w
75 4 w
99 2 w
100 11 r
111 60 r
```

   Nesse exemplo, as requisições com blocos iniciais 100, 106 e 101 foram fundidas em uma única requisição de 11 blocos, começando no 100. A requisição de bloco inicial 99 não pôde ser fundida por ser de escrita. Se a requisição de bloco inicial 111 fosse fundida com a de bloco inicial 106 (são adjacentes), a requisição resultante extrapolaria o limite de 64 blocos.