



ALUNO: Dênio Baines

1) (2,0 Pontos) Sobre conceitos básicos de sistemas operacionais, responda:

- 2.0 a) Descreva os estados de um processo e diga em que situações ocorrem as transições entre estes estados.  
b) O que são processos CPU-bound e I/O-bound? Exemplifique cada um deles.

2.0 2) (2,0 Pontos) Um sistema recebe a série de referências de páginas mostrada abaixo. O caractere "R" indica que a página foi acessada para leitura enquanto que o caractere "W" indica que a página foi acessada para escrita. A cada 12 páginas referenciadas, os bits R de todas as páginas são zerados pelo SO. O sistema tem cinco frames, que inicialmente encontram-se vazios. Para os algoritmos NUR e Segunda Chance, calcule a taxa de acerto e mostre o estado final memória real.

7R-3R-5R-2W-5W-5R-1R-3W-6R-4R-6R-4W-1R-4R-5R-3R-1R-2W-6R-1W

3) (2,0 Pontos) Sobre Entrada/Saída, responda:

- 1.0 a) O que são dispositivos mapeados em memória?  
b) Porque um sistema de E/S deve criar uma interface padronizada com os *device drivers*?

4) (2,0 Pontos) Considere um arquivo atualmente consistindo em 150 blocos. Suponha que o sistema de arquivos já esteja carregado na memória principal. Considere que existe espaço para crescimento no início e no final do arquivo. Suponha também que as informações de bloco a serem acrescentadas estejam armazenadas na memória principal. Calcule quantas operações de E/S (leitura e/ou escrita) de disco são necessárias para as estratégias de alocação contígua e lista ligada para cada uma das situações abaixo:

- a) Um bloco é acrescentado no início do arquivo.  
b) Um bloco é acrescentado entre o 57º e o 58º bloco do arquivo.  
c) Um bloco é acrescentado no final do arquivo

2.0 5) (2,0 Pontos) *Problema do Jantar Comunal*: Suponha que um grupo de  $N$  canibais come jantares a partir de uma grande travessa que comporta  $M$  porções. Quando alguém quer comer, se serve da travessa (apenas uma porção por vez), a menos que ela esteja vazia. Se a travessa está vazia, o canibal acorda o cozinheiro e espera até que o cozinheiro coloque mais  $M$  porções na travessa. A solução deve evitar deadlock e deve acordar o cozinheiro apenas quando a travessa estiver vazia. (Suponha um longo jantar, onde cada canibal continuamente se serve e come, sem se preocupar com as demais ações na vida do canibal...). Existe algum erro nos algoritmos? Se existir, explique detalhadamente os problemas ocasionados por este(s) erro(s) e em seguida faça as devidas correções.

SEMAPHORE COZINHAR = 1;

SEMAPHORE COMER = 0;

INT PORCAO = 0;

PROCESSO COZINHEIRO:

```
WHILE (TRUE) {  
    DOWN(COZINHAR);  
    PRODUZ_PORCAO();  
    PORCAO++;  
    IF (PORCAO == M) UP(COMER);  
    UP(COZINHAR);  
}
```

*IF (PORCAO == M) {  
 UP(COMER);  
 DOWN(COZINHAR);  
}*

BOA PROVA!

PROCESSO CANIBAL:

```
WHILE (TRUE) {  
    DOWN(COMER);  
    RETIRA_PORCAO();  
    PORCAO--;  
    IF (PORCAO == 0) UP(COZINHAR);  
    COME_PORCAO();  
    UP(COMER);  
}
```

*IF (PORCAO == 0) {  
 COME\_PORCAO();  
 UP(COZINHAR);  
 DOWN(COMER);  
 UP(COMER);  
}*







SEGUNDA CHANCE:

FALTAS: ☒ ☒  
ACERTOS: ☒ ☒

|   |           |
|---|-----------|
| 0 | 7 → 6 → 2 |
| 1 | 3 → 4     |
| 2 | 5 → 6     |
| 3 | 2 → 3     |
| 4 | 1         |

R: 1 1 1 1 1

7 3 5 2 1

R:

1 1 1 1  
5 2 1 6 4

→ R: 1 1 1 1  
3 1 2 4 6

5 2 1 6 4

1 1 1 1  
2 1 6 4 5 3

1 1 1 1  
4 5 3 1 2

50% de acertos MEM. = 2 4 6 3 1.

③ b) Para que haja uma mobilidade entre os dispositivos de I/O e os SOs, fez-se necessária a criação de uma técnica que fornecesse uma solução para a questão da compatibilidade de um determinado dispositivo com diferentes SOs. A solução foi criar uma interface padronizada entre a entrada e saída e o SO, esta interface interpreta os dados nos dois transmissores e os "traduz" para que tanto o SO compreenda a linguagem do dispositivo de I/O, como o dispositivo compreenda o SO.

④ a) alocação contígua: 1 leitura do bloco a ser escrito e 1 escrita. 2 operações.  
Ista ligada: 1 leitura do bloco a ser escrito e 1 escrita. 2 operações.

b) alocação contígua: como os dados estão distribuídos de forma seqüencial para necessário abrir um "buraco" para armazenar o bloco entre o 57° e 58°.



Assim os ~~deslizes~~ o bloco 150 passará a 151, 149 a 150.  
 e assim ~~para~~ diante. Assim terão de ser lidos os blocos  
 de 58 a 151 e haverá escrita nestes mesmos blocos. Perfazendo  
 93 leituras e 93 escritas. 186 operações de I/O.  
lista encadeada ligada: terão de ser lidos os 57 primeiros blocos  
 e haverá uma escrita. 58 operações de I/O.  
 OUV

c) alocação contígua: leitura do bloco 151 e escrita no  
 mesmo bloco. 2 operações de I/O.

alocação lista ligada: 150 leituras dos blocos anteriores  
 e 1 leitura e escrita do último bloco. 152 operações  
 escrita

⑤ De acordo com a sequência  $IF (PORCHO == M) U(COMER);$   
 $UP(COZINHAR);$

o cozinheiro continuará cozinhando enquanto os canibais  
 estão comendo, o que não deve ocorrer.  
 Substituí-lo então por:

```
IF (PORCHO == M) {
    UP(COMER);
    DOWN(COZINHAR);
}
UP(COZINHAR);
```

que garante que o cozinheiro só será acordado quando  
 $PORCHO == 0$ .

No processo canibal a sequência

```
IF (PORCHO == 0)
    UP(COZINHAR);
COMB-PORCHO(1);
UP(COMER);
```

deixa margem para que o cozinheiro comece a cozin-  
har enquanto ~~to~~ um canibal está comendo e o canibal retira  
uma porção que não existe.

Substituí-se por:

```
if (porção == 0) {  
    COMER_PORÇÃO();  
    UP(COZINHEIRO);  
    DOWN(COMER);  
    UP(COMER);  
}
```