Universidade Federal do Espírito Santo – Departamento de Informática
SP II (INF02828) & SO (INF02780) - Período: 2008/1
Primeira Prova Parcial– 20/05/2008

Profª Patrícia Dockhorn Costa
Email: pdcosta@inf.ufes.br

Regras importantes: A prova é sem consulta. Em caso de cola, a prova vale zero.

Questão 1) Considere um sistema que deva executar muitos tipos de processos, incluindo grandes, pequenos, com uso intensivo da UCP, com uso intensivo de E/S, em lotes e interativos. Considere ainda os seguintes algoritmos de escalonamento vistos em sala de aula: FIFO, SJF, SJF Preemptivo (ou SRTF), RR, Prioridades, Multinível. Dada a lista de requisitos a seguir, explique qual o algoritmo que melhor satisfaz a todos eles simultaneamente (explique como este algoritmo garante cada um dos requisitos). Para cada um dos demais algoritmos, indique pelo menos um requisito que não foi satisfeito e por quê. **(2,0)**

Requisito a) O número de processos interativos é relativamente pequeno. Eles deverão ter alta prioridade, mas são aceitáveis pequenos atrasos.

Requisito b) Processos com uso intensivo de E/S devem ter prioridade alta para manter ativos os processadores de E/S.

Requisito c) Processos com uso intensivo da UCP devem retardar processos com uso intensivo de E/S, mesmo que esses já estejam esperando há muito tempo.

Requisito d) Quando os processos com uso intensivo da UCP são os únicos que estão prontos, a sobrecarga do sistema operacional deve ser minimizada.

Questão 2) Suponha que os processos abaixo cheguem simultaneamente e de imediato passem a competir pela CPU. Considere ainda as seguintes abordagens de escalonamento: SJF preemptivo, FCFS e Round Robin. **(2,0)**

Process ID	CPU Burst (ms)
1	6
2	2
3	4
4	8

a) Qual é o turnaround time de cada processo em cada um dos algoritmos acima?
b) Qual é o waiting time de cada processo em cada um dos algoritmos acima?

Questão 3) Compare e contraste a manipulação de chamadas ao supervisor (SVCs) e interrupções em um sistema operacional. **(2,0)**

Questão 4) Considere a seguinte definição de semáforo: um semáforo S é uma variável inteira que pode ser acessada apenas por duas operações indivisíveis (atômicas): **(2,0)**

```
DOWN(S):        while (S ≤ 0); /* do nothing */
                S = S - 1;

UP(S):          S = S + 1;
```

Uma solução para o problema da exclusão mútua, usando essas operações, é mostrada abaixo:

```
semaphore S = 1;  /* S is initialized to 1 */
```

Cada processo executa o seguinte código:

```
while (TRUE) {
    DOWN(S);
    <in critical section>
    UP(S);
}
```

(a) A solução acima envolve algum *busy-waiting*? Explique.
(b) O que é o problema da "inversão de prioridade"?
(c) Se o escalonamento por prioridade é usado, a solução acima pode levar ao problema da inversão de prioridade? Se sim, dê um exemplo; do contrário, explique porque não.

Questão 5) Suponha que um grupo de N canibais come jantares a partir de uma grande travessa que comporta M porções. Quando alguém quer comer, ele(ela) se serve da travessa, 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. Utilizando semáforos, desenvolva o código para sincronizar as ações dos canibais e do cozinheiro. A solução deve evitar *deadlock* e 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 atividades na vida de um canibal. **(2,0)**

BOA PROVA!