UFES - DEPARTAMENTO DE INFORMÁTICA

2ª. Prova de Sistemas Operacionais / Sistemas de Programação II

2011/2 – Profa. Roberta Lima Gomes

1) **(2,0)** No primeiro trabalho prático da disciplina, a seguinte especificação foi passada:

"... Outra particularidade da `fsh` é que uma família de processos é muito unida! Quando um dos processos morre, todos os demais processos da família morrem juntos, à exceção da `fsh`. Esse genocídio de processos deve ser implementado com o auxílio de sinais (alterando-se a rotina de tratamento de um determinado sinal, um processo que está para morrer deve enviar sinais para matar outros processos de sua família). "

Explique qual foi o problema encontrado para implementar a especificação acima. Como o seu grupo tratou esse problema? A solução do seu grupo apresenta alguma limitação (critique essa solução)?

2) **(3,0)** Três tipos de processos compartilham acesso a uma lista simplesmente encadeada: *searchers*, *inserters* e *deleters*. Os *searchers* apenas examinam a lista realizando buscas, logo podem executar concorrentemente entre si. Os *inserters* devem ser mutuamente exclusivos para evitar inserções paralelas. No entanto, uma inserção pode prosseguir em paralelo com qualquer número de buscas. Finalmente, os *deleters* podem ter acesso à lista um de cada vez e devem ser mutuamente exclusivos com buscas e inserções. Escreva o código para os três tipos de processos que assegurem estas propriedades. (Utilize sintaxe de uma linguagem ou pseudo-código)

3) **(2,0)** Considere a seguinte modelagem, por monitor, para o problema do produtor-consumidor com *n* produtores e *m* consumidores.

```
monitor buffer {
  int itens=0; cond temItens, temEspacos;
  ...
  int pega() {
    while (1) {
      if (!itens) wait(temItens);
      pega item no buffer
      itens--;
      signal(temEspacos);
      return (item);}
  }
  void coloca() {
    while (1) {
      if (itens==MAX) wait(temEspacos);
      coloca item no buffer
      itens++;
      signal(temItens);}
  }
}
```

Suponha que esse monitor funcione com a disciplina "sinaliza e continua". Explique por que essa solução não funciona corretamente. Mostre uma sequência de ações que ilustre o problema. Que outra disciplina poderia ser usada de forma que esta solução estaria correta?

4) **(1,5)** Mostre a equivalência de funcionalidade entre semáforos e o mecanismo de troca de mensagens. Use um vetor localizado em memória compartilhada no papel de "caixa postal" para mostrar como implementar semáforos usando troca de mensagens.

5) **(1,5)** Diferencie *user-level threads* de *kernel-level threads*, explicitando suas vantagens e desvantagens . No Linux é possível utilizar qual tipo de threads? Explique.

Boa prova!!