Curso de Tecnologia em Sistemas de Computação
Disciplina de Sistemas Operacionais
Professores: Valmir C. Barbosa e Felipe M. G. França
Assistente: Alexandre H. L. Porto

Quarto Período
AP1 - Primeiro Semestre de 2009

Nome -
Assinatura -

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

---

1. (2.0) Para os usuários do computador, qual é a diferença essencial entre os sistemas de lote e os sistemas de compartilhamento de tempo?

   Resp.: A diferença é que em um sistema de lote, os usuários submetem os seus programas para execução no computador. Posteriormente, os programas de diversos usuários são organizados em lotes pelos operadores, e são colocados, por estes, para executarem no computador. Após as execuções, os resultados dos programas são coletados pelos operadores (por exemplo, eles são impressos), e são disponibilizados aos usuários que originalmente submeteram os programas. Em essência, não existe nenhuma interação do usuário com o computador, somente do usuário com os operadores. Já em um sistema de compartilhamento de tempo, os usuários acessam o computador a partir de terminais, portanto interagindo diretamente com o computador. Além disso, os próprios usuários, que têm a ilusão de estarem usando exclusivamente o computador, colocam os programas para executarem no computador e coletam os resultados destas execuções após o término dos programas.

2. (1.5) Qual é a principal vantagem de uma hierarquia de diretórios em relação a uma abordagem alternativa em que todos os arquivos do sistema operacional sejam armazenados em um mesmo diretório?

   Resp.: Quando todos os arquivos são armazenados em um mesmo diretório, além de não poderem existir dois ou mais arquivos com mesmo nome, também não se podem organizar os arquivos, em categorias, de acordo com os seus conteúdos. Agora, se o sistema operacional possui uma hierarquia de diretórios, podem existir dois ou mais arquivos com o mesmo nome, desde que sejam armazenados em diretórios diferentes. Além disso, a própria hierarquia, em que os diretórios de um nível, diferente do primeiro, são subdiretórios do diretório do nível imediatamente anterior (o diretório do primeiro nível, o diretório raiz, não é subdiretório de nenhum diretório), permite naturalmente organizar os arquivos de acordo com os seus conteúdos. Por exemplo, em uma hierarquia de diretórios, cada usuário do sistema operacional pode ter o seu próprio diretório, e pode criar, por exemplo, um subdiretório para figuras, um para textos e um para músicas. Se existisse somente um

diretório, não somente todos os arquivos de todos os usuários deveriam ser armazenados nele, como um usuário não poderia criar subdiretórios e dividir os seus arquivos de acordo com os seus conteúdos.

3. (1.5) A multiprogramação precisa ser visível ao usuário do computador? Justifique a sua resposta.

   Resp.: Não, a multiprogramação não precisa ser visível ao usuário do computador. Se o sistema operacional executar no hardware e implementar a multiprogramação, então ela será visível ao usuário do computador. Isso ocorrerá porque ele saberá que cada programa é executado por um processo diferente do sistema operacional, e que estes processos compartilham o(s) processador(es) do hardware. Agora, se o sistema operacional executar em uma máquina virtual, o usuário do computador não saberá que o processador do hardware está sendo multiprogramado, pois terá a impressão de que o sistema operacional está executando exclusivamente no computador. Quem será o responsável por multiprogramar o processador, neste caso, será o monitor de máquina virtual, que irá mapear, de tempos em tempos, o processador de cada máquina virtual no processador do hardware. Note que o usuário saberá, porém, da multiprogramação do processador da máquina virtual, se o sistema operacional for multiprogramado.

4. (1.5) Como o modelo de processos facilita o entendimento e a implementação da multiprogramação?

   Resp.: No modelo de processos, o sistema operacional é visto como um conjunto de processos sequenciais. Para cada processo, as informações necessárias à sua execução, incluindo os registradores do processador, são salvas no seu contexto quando ele deixa de executar no processador, e são restauradas quando ele voltar a executar no processador. Logo, podemos imaginar que cada um destes processos está executando em um processador virtual cujo conteúdo dos registradores são exatamente aqueles que foram salvos no contexto do processo. Portanto, o conceito facilitará o entendimento da multiprogramação, pois ela ocorrerá naturalmente quando alternarmos o uso do processador do hard-

ware entre estes processadores virtuais. O conceito também facilitará a implementação da multiprogramação, pois bastará salvarmos os registradores do processador no contexto do processo quando ele deixar de executar no processador, e restaurar os registradores do contexto imediatamente antes de o processo voltar a executar no processador.

5. (1.5) Descreva como os semáforos são implementados no sistema operacional, destacando o modo de funcionamento das operações **P** e **V**.

   Resp.: Cada semáforo do sistema operacional é composto por um valor inteiro e por uma fila que armazena as informações sobre cada processo bloqueado por causa deste semáforo. As operações **P** e **V** são implementadas no núcleo do sistema operacional de modo atômico, isto é, os códigos que as implementam são executados do início até o fim sem interrupções, para garantir a exclusão mútua ao acessar o valor e a fila. A ação executada pela operação **P** dependerá do valor associado ao semáforo. Se este valor for maior do que 0, a operação simplesmente o decrementará em uma unidade. Porém, se este valor for igual a 0, o processo que executou a operação será bloqueado, e as informações que o identificam serão inseridas na fila do semáforo. A ação executada pela operação **V** também dependerá do valor do semáforo. Se o valor for maior do que 0, ele será incrementado em uma unidade se o semáforo for de contagem (pois o valor de um semáforo binário somente pode ser 0 ou 1). Se o valor for 0 e a fila não estiver vazia, um dos processos cujas informações estão na fila será escolhido aleatoriamente, as suas informações serão removidas da fila, e ele será desbloqueado e colocado no estado **pronto**, para depois poder ser escolhido pelo escalonador. Finalmente, se o valor for 0 e a fila estiver vazia, ele também será incrementado em uma unidade.

6. (2.0) O algoritmo de escalonamento por *round robin* precisa ser preemptivo, ou pode ser também não-preemptivo? Justifique a sua resposta.

   Resp.: O algoritmo precisa ser preemptivo, como veremos a seguir. Como vimos na Aula 6, no algoritmo de escalonamento por *round robin*, cada processo executa no processador por no máximo um *quantum*

(um intervalo fixo de tempo definido pelo sistema operacional). Após executar por um *quantum*, o processo somente executará novamente, por mais um *quantum*, após todos os outros processos no estado **pronto** executarem também por um *quantum*. Para garantir o funcionamento do algoritmo, precisamos executar o escalonador em intervalos fixos de tempo (iguais ao *quantum*), para que ele possa alternar o uso do processador entre os processos no estado **pronto**. Isso não poderá ser feito se o algoritmo for não-preemptivo porque, neste caso, o escalonador somente será executado se o processo atualmente em execução fizer alguma operação de E/S ou terminar a sua execução. Já um algoritmo preemptivo, além de executar o escalonador nestes casos, o executa também em intervalos fixos de tempo, usando o temporizador do hardware para gerar uma interrupção entre cada um destes intervalos. Logo, para poder alternar o uso do processador entre os processos em intervalos fixos de tempo iguais ao *quantum*, precisaremos que o algoritmo seja preemptivo.