Atividade de Sistemas Operacionais

Parte 1 (10 Questões)

Questão 4 (Capítulo 13 - Impasses)

Eliminar tarefas: uma ou mais tarefas envolvidas no impasse são eliminadas, liberando seus recursos para que as demais tarefas possam prosseguir. A escolha das tarefas a eliminar deve levar em conta vários fatores, como o tempo de vida de cada uma, a quantidade de recursos que cada tarefa detém, o prejuízo para os usuários que dependem dessas tarefas, etc.

Retroceder tarefas: uma ou mais tarefas envolvidas no impasse têm sua execução parcialmente desfeita (uma técnica chamada *rollback*), de forma a fazer o sistema retornar a um estado seguro anterior ao impasse. Para retroceder a execução de uma tarefa, é necessário salvar periodicamente seu estado, de forma a poder recuperar um estado anterior quando necessário.

Questão 8 (Capítulo 13 - Impasses)

Grafo 1: t2 → r1 → t1 → r2 → t2
Grafo 2: t3 → r1 → t1 → r2 → t2 → r3 → t3

Questão 9 (Capítulo 13 - Impasses)

Exclusão mútua: a via deve ser, ao menos de forma momentânea, exclusiva para um veículo. Assim, somente um carro pode acessar por vez;

Posse e espera: os carros possuem uma via, porém, esperam o acesso para outra;

Não-preempção: apenas o veículo pode liberar acesso à via, já que a via alocada para o veículo não pode ser removida.

Espera circular: um carro requer a via para outro veículo.

Uma boa estratégia para a resolução do impasse seria a *rollback*; retornando a um estado anterior facilitaria a ordenação dos recursos usados para os veículos.

Questão 1 (Capítulo 14 - Gestão de Memória)

Edição: o programador escolhe o endereço de cada uma das variáveis e do código do programa na memória. Esta abordagem normalmente só é usada na programação de sistemas embarcados simples, programados diretamente em Assembly.

Compilação: ao traduzir o código-fonte, o compilador escolhe as posições das variáveis na memória. Para isso, todos os códigos-fontes necessários ao programa devem ser conhecidos no momento da compilação, para evitar conflitos de endereços entre variáveis em diferentes arquivos ou bibliotecas. Essa restrição impede o uso de bibliotecas pré-compiladas. Esta abordagem era usada em programas executáveis com extensão *.COM* do *MS-DOS* e *Windows*.

Ligação: na fase de compilação, o compilador traduz o código fonte em código binário, mas não define os endereços das variáveis e funções, gerando como saída um arquivo objeto (*object file*) 2 , que contém o código binário e uma tabela de símbolos descrevendo as variáveis e funções usadas, seus tipos, onde estão definidas e onde são usadas. A seguir, o ligador (*linker*) pega os arquivos objetos com suas tabelas de símbolos, define os endereços de memória dos símbolos e gera o programa executável.

Carga: também é possível definir os endereços de variáveis e de funções durante a carga do código em memória para o lançamento de um novo processo. Nesse caso, um carregador (*loader*) é responsável por carregar o código do processo na memória e definir os endereços de memória que devem ser utilizados. O carregador pode ser parte do núcleo do sistema operacional ou uma biblioteca ligada ao executável, ou ambos. Esse mecanismo normalmente é usado na carga de bibliotecas dinâmicas (*DLL*).

Execução: os endereços emitidos pelo processador durante a execução do processo são analisados e convertidos nos endereços efetivos a serem acessados na memória real. Por exigir a análise e a conversão de cada endereço gerado pelo processador, este método só é viável com o auxílio do hardware.

Questão 2 (Capítulo 14 - Gestão de Memória)

TEXT: contém o código binário a ser executado pelo processo, gerado durante a compilação e a ligação com as bibliotecas e armazenado no arquivo executável. Esta seção se situa no início do espaço de endereçamento do processo e tem tamanho fixo, calculado durante a compilação.

DATA: esta seção contém as variáveis estáticas inicializadas, ou seja, variáveis que estão definidas do início ao fim da execução do processo e cujo valor inicial é declarado no código-fonte do programa. Esses valores iniciais são armazenados no arquivo do programa executável, sendo então carregados para esta seção de memória quando o processo inicia. Nesta seção são armazenadas tanto variáveis globais quanto variáveis locais estáticas (por exemplo, declaradas como *static* em C).

BSS: historicamente chamada de *Block Started by Symbol*, esta seção contém as variáveis estáticas não-inicializadas. Esta seção é separada da seção DATA porque as variáveis inicializadas precisam ter seu valor inicial armazenado no arquivo executável, o que não é necessário para as variáveis não-inicializadas. Com essa separação de variáveis, o arquivo executável fica menor.

HEAP: esta seção é usada para armazenar variáveis alocadas dinamicamente, usando operadores como ***malloc***(), ***new***() e similares. O final desta seção é definido por um ponteiro chamado *Program Break*, ou simplesmente break, que pode ser ajustado através de chamadas de sistema para aumentar ou diminuir o tamanho da mesma.

STACK: esta seção é usada para manter a pilha de execução do processo, ou seja, a estrutura responsável por gerenciar o fluxo de execução nas chamadas de função e também para armazenar os parâmetros, variáveis locais e o valor de retorno das funções. Geralmente a pilha cresce “para baixo”, ou seja, inicia em endereços maiores e cresce em direção aos endereços menores da memória. O tamanho total desta seção pode ser fixo ou variável, dependendo do sistema operacional. Em programas com múltiplas *threads*, esta seção contém somente a pilha do programa principal. Como *threads* podem ser criadas e

destruídas dinamicamente, a pilha de cada thread é mantida em uma seção de memória própria, alocada dinamicamente no *heap* ou em blocos de memória alocados na área livre para esse fim.

Questão 1 (Capítulo 15 - Hardware de Memória)

Endereços físicos (ou reais) são os endereços dos bytes de memória física do computador. Estes endereços são definidos pela quantidade de memória disponível na máquina.

Endereços lógicos (ou virtuais) são os endereços de memória usados pelos processos e pelo sistema operacional e, portanto, usados pelo processador durante a execução. Estes endereços são definidos de acordo com o espaço de endereçamento do processador.

Questão 7 (Capítulo 15 - Hardware de Memória)

Fórmula: *Physical Address = Base Ad + Offset;*
Offset < Limit

- 0:45 = 89;
- 1:100 = 300;
- 2:90 = 90;
- 3.1900 = 3900; INVÁLIDO (1900 > 1000 *LIMIT*)
- 4:200 = 1400.

Questão 8 (Capítulo 15 - Hardware de Memória)

Fórmula: *(Physical Frame * Page Size) + Offset*
physalF = logicalA/pageS (divisão inteira)
Offset = logicalA%PageS

- 414 = 1914;
- 741 = 741;
- 1995 = 495; (*NULL* - não tem frame referente)
- 4000 = 0;
- 6633 = 6633.

Questão 5 (Capítulo 16 - Alocação de Memória)

(b) Se usarmos *Worstfit*, o tamanho final do buraco B4 será de 15 *Mbytes*.

Questão 1 (Capítulo 17 - Paginação em Disco)

É a ausência de página na memória RAM verificada pela MMU, gerando assim, uma interrupção que desvia a execução para o sistema operacional. O sistema operacional verifica se a página é válida, usando os flags de controle da tabela de páginas. Caso a página seja inválida, o processo tentou acessar um endereço inválido e deve ser abortado. Por outro lado, caso a página solicitada seja válida, o processo deve ser suspenso enquanto o sistema transfere a página de volta para a memória RAM e faz os ajustes necessários na tabela de páginas.

Uma vez a página carregada em memória, o processo pode continuar sua execução.