Sistemas Operacionais - Aula 2

Nome: Maurício Witter - 201911351

1 - Por que o código-objeto gerado pelo tradutor ainda não pode ser executado?

Um tradutor é um “compilador” genérico que tem apenas como função traduzir um código fonte para um código de baixo nível. Ou seja, ele traduz o código para um módulo-objeto que ainda não é um arquivo executável. Para executar este módulo-objeto, é necessário que um Linker e o Loader sejam usados para fazer o link das bibliotecas e carregar na memória.

2 - Por que a execução de programas interpretados é mais lenta que a de programas compilados?

Interpretadores de código se diferem dos compiladores em alguns pontos que faz com que eles sejam um pouco mais lentos. Código compilado se torna estático e praticamente imutável senão por patches do binário ou injeção de código no processor. Já código que é interpretado é mais flexível e pode ser alterado em *run time*, ou seja, o interpretador irá interpretar uma injeção de código sem reiniciar a aplicação inteira. O Problema disso é que o bytecode do interpretador não passou por um processo de otimização profunda e por um linker o que torna a interpretação mais lenta. Atualmente, muitos interpretadores adotaram um modelo de compilação (JIT - Just-in-Time) para otimizar o código e tornar o processo de interpretação mais rápido fazendo cache de binários mais acessados, por exemplo.

3 - Quais as funções do linker?

Na primeira fase de compilação, onde o compilador realiza as etapas de *lexer* e análise sintática, o compilador não realiza a tabulação predeterminada de endereços de memória, mas adiciona uma flag uma referência hexadecimal para o linker checar. Assim, o linker realiza o link desses módulos-objetos e irá resolver todas as referências simbólicas existentes entre os módulos-objeto, reservar memória para a execução do

programa e determinar uma região da memória onde o programa será carregado para sua execução. Além disso, ele realiza otimizações quando possível para remover código desnecessário visto que o código compilado será estático.

4 - Qual a principal função do loader?

Em ambientes multiprogramáveis onde não é possível predeterminar os endereços de memória de forma fixa como o linker faz, irá reservar endereços de acesso aleatório para carregar o programa na memória principal para execução. O loader carrega um programa para memória principal, ele aloca uma área de código, uma área de dados e uma área de pilha. A área de código armazena o programa executável, a área de dados armazena as variáveis e constantes utilizadas no programa e a área de pilha armazena os endereços de retorno das funções ou procedimentos chamados durante a execução do programa.

O Loader pode ser absoluto ou realocável dependendo do código gerado pelo *linker*. Quando absoluto, será necessário conhecer os endereços determinados pelo *linker* e carregar o programa de memória secundária para a principal. Quando realocável, o loader o programa pode ser carregado em qualquer posição de memória e o loader é responsável pela realocação no momento do carregamento

5 - Quais as facilidades oferecidas pelo depurador?

Um depurador (debugger) auxilia o programador a entender o código e o comportamento do programa durante a execução. Suas funções permitem executar o código linha por linha, colocando breakpoints para verificar o estado das variáveis e valores correntes em run-time ou compile-time. Com isso, se torna mais fácil para o programador encontrar bugs e corrigi-los.

6 - O que é concorrência e como este conceito está empregado nos SOs multiprogramáveis?

A concorrência é um conceito que permite que múltiplos programas sejam executados simultaneamente, ou seja, quando um está fazendo uma execução de I/O, outros podem continuar executando na CPU. Por exemplo, em um sistema operacional multiprogramável, o kernel é responsável por gerenciar as threads e os programas são executados nessas threads, assim vários programas são carregados na memória e executados de forma intercalada. A concorrência só é possível em sistemas multiprogramáveis, pois os recursos computacionais são compartilhados entre as aplicações e os programas concorrem por recursos podendo executar tarefas em multithread simultaneamente, dependendo da prioridade e outros critérios.

7 - Por que o mecanismo de interrupção é fundamental para a implementação da multiprogramação?

O mecanismo de interrupção em ambientes multiprogramáveis é um tipo de evento que ocorre quando um processo termina de executar ou acontece um erro que tira o programa do fluxo normal de execução. Quando o processo é concluido com sucesso o interruptor termina a execução normalmente, mas quando ocorre um erro o interruptor faz um desvio para uma rotina responsável por tratar o evento ocorrido, ou seja, para salvar as informações em registradores para que o programa retome o fluxo de execução caso seja possível corrigir o erro. Dessa forma, esse mecanismo é fundamental para interromper a execução de programas e desalocar recursos ou recuperar de um erro.

8 - O que são eventos síncronos e assíncronos?

Em sistemas multiprogramáveis, um evento assíncrono é independente dos dados de entrada e das instruções do programa, ou seja, pode ocorrer em qualquer ponto do programa. Essas interrupções não estão relacionadas com a instrução do programa corrente, são eventos imprevisíveis, podem ocorrer múltiplas vezes. Já um evento síncrono, é resultado direto da execução do programa corrente. Tais eventos são previsíveis, e se um mesmo programa for executado várias vezes com a mesma entrada de dados, os eventos síncronos ocorrerão sempre nos mesmos pontos do programa.

9 - Dê exemplos de eventos associados ao mecanismo de exceção.

A exceção provoca um erro fatal no sistema, causando o término anormal do programa. Assim, uma exceção é um evento que deve ser melhor tradado pelo próprio programa. Alguns exemplos de exceção são: Runtime Exception (Null Pointer Exception, Illegal Argument Exception, Index out of bound Exception) - IOException (File Not Found) - Out of memory Exception (Stack Overflow) - Stack Overflow Exception (Stack Overflow) - Interrupted Exception (Interrupted Exception) - Arithmetic Exception (Division by zero) etc.

10 - O que é DMA e qual a vantagem desta técnica ?

Direct Memory Access (DMA) permite que certos dispositivos de hardware num computador acessem a memória do sistema para leitura e escrita independentemente da CPU. Além do processador acessar a memória RAM, a DMA permite que outros componentes também acessem a memória RAM diretamente, como discos rígidos. Ou seja, em vez da CPU ter que copiar todos os dados da fonte até o destino, que é tipicamente mais lento do que copiar blocos de dados dentro da memória, já que o acesso a dispositivo de I/O através de barramentos periféricos é mais lento que a RAM. Além disso, durante a cópia dos dados a CPU ficaria indisponível para outras tarefas. A vantagem é que o uso típico do DMA ocorre na cópia de blocos de memória da RAM do sistema para um buffer do dispositivo. Estas operações não bloqueiam o processador que fica livre para realizar outras tarefas.

11 - Como a técnica de buffering permite aumentar a concorrência em um sistema computacional?

Buffer é uma região de memória física utilizada para armazenar temporariamente os dados enquanto eles estão sendo movidos de um lugar para outro. Os buffers normalmente são usados quando há uma diferença entre a taxa a qual os dados são

recebidos e a taxa a qual eles podem ser processados, ou no caso em que estas taxas são variáveis. Frequentemente ajusta o tempo pela implementação de um algoritmo de fila (FIFO) na memória, simultaneamente escrevendo dados na fila em uma taxa e lendo-os em outra taxa. Utilizando a concorrência, pode-se utilizar as threads para adicionar dados à fila em diferentes locais do buffer.

12 - Em um sistema multiprogramável, seus usuários utilizam o mesmo editor de textos (250Kb), compilador (320Kb), software de e-mail (230 Kb) e uma aplicação corporativa (570 Kb).

a) Sem a implementação de reentrância, qual o espaço de memória principal ocupado pelos programas quando 10 usuários estiverem utilizando todas as aplicações simultaneamente?

Sem a utilização de Reentrância, cada programa executado em cada usuário seria uma nova instância, assim no total a memória ocupada por 10 usuários seria 13.700 Kb.

b) Qual o espaço liberado quando o sistema implementa reentrância em todas as aplicações?

Dado que a memória é compartilhada entre os usuários, mesmo com 10 usuários utilizando os programas simultaneamente, o espaço liberado será de 12.330 KB.

REFERÊNCIAS

COUTINHO, C. B.. **Sistemas Operacionais,** 2010**.**

FRANCISCATTO, R.. **Tipos de Software e Concorrência de Recursos**.

TENENBAUM, A.. **SISTEMAS OPERACIONAIS MODERNOS**.