Resumo Prova 2 SO

Guilherme Christopher Michaelsen Cardoso *

14 de maio de 2017

*baseado no livrão do Tanenbaum

1 Resumão de SO (Prova 2)

1.1 Espaços de endereçamento

Expor memória física para processos tem diveras desvantagens:

- Permitir que programas de usuário enderecem qualquer byte de memória torna possível quebrar o sistema operacional.
- Com esse modelo, é dificil ter vários programas rodando ao mesmo tempo.

1.1.1 Noção de espaço de endereçamento

É necessário resolver dois problemas antes de permitir múltiplas aplicações em memória ao mesmo tempo: proteção e relocação.

- Espaço de Endereçamento:
  - Conjunto de endereços que um processo pode usar para endereçar memória.
  - Cada processo tem seu próprio espaço de endereçamento, independente dos outros processos (exceto em circunstâncias especiais onde processos querem compartilhar seus espaços de endereçamento).

1.2 Memória Virtual

- Enquanto a capacidade das memórias cresce rapidamente, o tamanho dos programas cresce ainda mais rápido.
- Necessidade de rodar programas que são grandes demais para caber em memória

- Swapping não é uma alternativa desejável, devido à lentidão dos discos rígidos.
- **Memória virtual** foi a solução encontrada para esse problema.
  - Cada programa tem seu próprio espaço de endereçamento, que é dividido em partes chamadas **páginas**.
  - Cada página é um intervalo contíguo de endereços, que são mapeados em memória física.
  - Nem todas as páginas precisam estar em memória física ao mesmo tempo para rodar o programa.
  - Quando o programa referencia uma parte do endereço que já está em memória física, o hardware realiza o mapeamento necessário na hora.
  - Quando o programa referencia uma parte do espaço de endereçamento que não está em memória, o SO recebe um aviso para buscar a página faltante e reexecutar a instruçao que falhou.
  - Funciona muito bem em sistemas multiprogramados com pedaços de vários programas na memória ao mesmo tempo. Quando um programa está esperando uma página ser buscada do disco, a CPU pode ser concedida a outro processo.

1.2.1 Paginação

- A maioria dos sistemas de memória virtual usam **paginação**
- Programas geram **endereços virtuais** de memória, formando o **espaço de endereçamento virtual**.
- Em sistemas com memória virtual, esses endereços não são enviados diretamente ao barramento da memória. Ao invés disso, vão para uma **MMU (Memory Management Unit)** que mapeia o endereço virtual em endereços de memória reais.

- O espaço de endereçamento virtual consiste de unidades de tamanho fixo chamadas **páginas**. AS unidades correspondentes em memória física são chamadas de **molduras de página**. As páginas e molduras de página geralmente tem o mesmo tamanho.
- Supondo um sistema de memória com páginas de 4KB, 64KB de espaço de endereçamento virtual e 32KB de memória física, temos 16 páginas virtuais e 8 molduras de páginas.
- Sempre que se necessita buscar um item no disco, é necessário que a página inteira seja buscada.
- Vários processadores suportam múltiplos tamanhos de página que podem ser misturados pelo SO de acordo com a necessidade.
  - Ex.: A arquitetura x86_64 suporta páginas de 4KB, 2MB e 1GB, sendo possível, por exemplo, utilizar páginas de 4KB para aplicações de usuário e uma única página de 1GB para o kernel do SO.
- Exemplo: Supondo páginas de 4KB, 64KB de espaço de endereçamento virtual e 32KB de memória física. Um programa solicita um dado que está no endereço virtual 0. Supondo que a página que contém os endereços 0 a 4095 está mapeada na moldura de página numero 2. Qual o endereço de memória real que será emitido pela MMU?
  - A moldura 0 contém os endereços 0 a 4095. A moldura 1 contém os endereços 4096 à 8191. Logo, a moldura 2 começa no endereço 8192 e vai até 12287. Portanto, o endereço físico que será emitido será 8192.
- Exemplo 2: O endereço virtual 20500 é 20 bytes à partir do começo da página virtual 5 (endereços virtuais 20480 a 24575).

Supondo que essa página seja mapeada para a moldura de página número 3 (12k-16k), o endereço físico emitido pela MMU será 12288 + 20 = 12308.

- Como existem mais páginas virtuais do que moldura física, é necessário controlar quais páginas virtuais estão presentes em memória física. Para isso, é usado um bit de controle (**bit presente/ausente**).
- Caso o programa referencie um endereço não mapeado, a MMU detecta isso e faz com que a CPU interrompa o sistema operacional. Essa interrupção é chamada de **page fault**. O SO então escolhe uma moldura de memória pouco usada, salva seu conteúdo no disco (caso já não esteja lá), e então busca (também do disco) a página que acabou de ser referenciada, colocando-a na moldura de página que foi liberada, refaz o mapeamento, e reinicia a instrução interrompida.
- Exemplo: supondo um endereço virtual 8196 (0010000000000100 em binário), sendo mapeado pela MMU. Esse endereço virtual de 16 bits é dividido em duas partes: um número de página (4 bits) e um offset (12 bits). Com 4 bits de offset, podemos endereçar $2^4 = 16$ páginas, e com um offset de 12 bits, podemos endereçar todos os $2^{12} = 4096$bytes em uma página.