Gestão de memória

Licenciatura em Tecnologias e Sistemas de Informação

Sistemas Operativos

Helena Rodrigues
(helena@dsi.uminho.pt)

Gestão de memória

Bibliografia:

Silberschatz, A., Galvin, P., Gagne, Greg, *Applied Operating Systems Concepts*, Addison Wesley, 2000, cap. 9

Edição pdf: cap. 8

Gary Nutt, *Operating Systems*, Third edition, Addison Wesley, 2004, ISBN 0-321-18955-8, cap. 11

Gestão de memória

Resultados de aprendizagem:

- Explicar a função e objectivos do gestor de memória
- Explicar a diferença entre espaço de endereçamento lógico e espaço de endereçamento físico
- Explicar e comparar as técnicas de atribuição de memória baseadas em páginas - paginação e em segmentos - segmentação
- Explicar as diferentes estratégias de gestão de memória, incluindo a memória virtual e a sua influência no desempenho do sistema de computação.

Gestão de memória

- Sistema de Memória
  - memória secundária (discos duros)
    - armazena grandes quantidades de informação
    - informação sobrevive à execução de um programa
    - meio de partilha de informação
    - Guarda ficheiros binários correspondentes a programas
  - memória principal
    - execução de um programa (processo) => programa e dados na memória principal
  - Para serem executados os programas são carregados para a memória principal!

Gestão de Memória

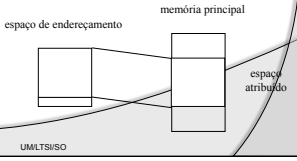
- Enquadramento
  - Processos precisam de memória para executar
  - Vários processos em execução concorrente
  - Necessário gerir a utilização da memória pelos vários processos em execução
- Sistema de gestão de memória
  - Atribui a memória disponível aos vários processos que dela necessitam para poderem ser executados
- Questão: Indique algumas das questões que com que um sistemas de gestão de memória terá de lidar
  - Exemplo: Dado um processo para ser executado em que área do espaço de memória colocar esse processo?

Gestão de Memória

- Em que área de memória colocar cada programa?
- Como é que os programas vão saber o espaço de memória que têm disponível?
- Como evitar que um processo manipule o espaço de memória atribuído a outro processo?
- Como é que os programas vão saber os endereços de memória com que vão trabalhar para poderem referir-se a essa mesma memória?
- Como permitir que vários programas sejam executados concorrentemente mesmo que não caibam todos ao mesmo tempo na memória disponível?
- Como permitir que um programa execute estando apenas em parte na memória principal?
- Como partilhar memória entre vários utilizadores para permitir por exemplo que um programa usado por vários utilizadores possa usar apenas uma única área de memória?

Gestão de Memória

- Tarefa do Gestor de Memória
  - atribuir e libertar memória para a execução de processos
    => transferir automaticamente programas (código e dados) de memória secundária para memória principal e vice-versa
  - minimizar os tempos de acesso à memória usando uma "razoável" quantidade de memória
  - Associar o espaço de endereçamento dos processos à porção de memória *atribuída*



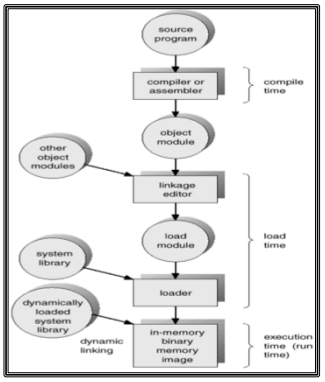
Gestão de Memória

- Todas as referências a variáveis e a instruções do programa em execução terão em algum ponto de ser associadas a uma determinada posição de memória
- Exemplos
  - Executar a próxima instrução
    - *Fetch* da próxima instrução indicada pelo *Program Counter*
  - Atribui um valor a uma variável
    - Associação de endereços em memória

```
(cont=1  => STORE   8CA87,1)
```


Gestão de Memória

- A associação dos endereços das instruções e dados a endereços de memória pode ser feita em:
  - **Tempo de compilação**: se a posição de memória a atribuir é conhecida antecipadamente, é possível gerar endereços absolutos; se a posição é alterada, o código tem de ser recompilado.
  - ***Load time***: se a posição de memória a atribuir não é conhecida em tempo de compilação, o compilador gera código re-atribuível (*relocatable*); quando o programa é carregado em memória os endereços são ajustados.
  - **Tempo de execução**: se durante a execução de um programa, este pode alterar a sua posição em memória, a associação de endereços é adiada para o tempo de execução.

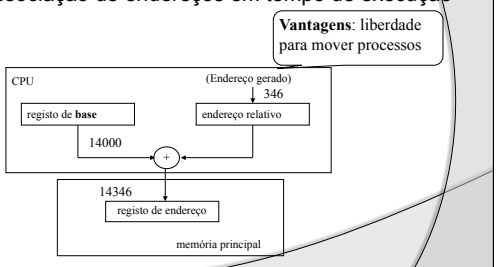


Gestão de Memória

- Associação de endereços em memória em tempo de execução
  - Abordagem mais comum
  - Grande vantagem é facilitar a movimentação de processos na memória
  - os endereços são ajustados pelo hardware em tempo de execução
    - registo de **Base**
    - registo de **Limite**

Gestão de Memória

- Associação de endereços em tempo de execução



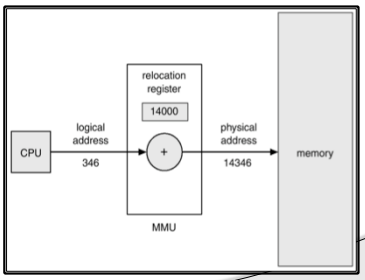
Gestão de Memória

- Espaço de endereçamento lógico *vs* físico
  - O espaço de endereçamento lógico é associado a um espaço de endereçamento físico independente.
    - Endereço lógico – endereços referidos no código; também denominado por endereço virtual.
    - Endereço físico – endereço visível à unidade de memória.
  - O espaço de endereçamento lógico e o espaço físico são o mesmo quando a associação de endereços é feita em tempo de compilação ou em *load time*; são diferentes quando a associação de endereços é feita em tempo de execução.

Gestão de Memória

- *Memory Management Unit* (MMU)
  - Elemento de hardware que suporta o mapeamento/ associação entre endereços lógicos e físicos
  - Existe um endereço de base que é adicionado aos endereços lógicos gerados pelo CPU para obter o correspondente endereço físico.
  - O programa apenas lida com endereços lógicos nunca tendo de lidar com endereços físicos

Gestão de Memória



Questões

- Qual é que poderá ser maior?
  - O espaço de endereçamento físico
  - O espaço de endereçamento lógico
- Para poder ser executado um programa precisa de ser carregado para a memória principal mas é possível permitir que:
  - Um processo possa sair de memória quando não estiver no estado de execução e de seguida ser novamente carregado
  - Apenas uma parte do programa esteja em memória
- Quais serão as vantagens de poder executar um programa sem que este tenha de estar sempre e completamente carregado em memória?

Gestão de Memória

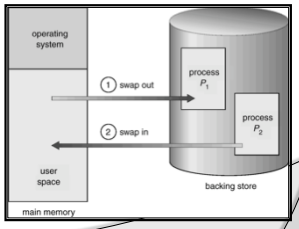
- Sistemas de Gestão de Memória com permutação
  - ***Swapping***
  - **Memória Virtual**

Gestão de Memória

- *Swapping*
  - é adequado a sistemas multi-utilizador
  - remove processos bloqueados da memória principal para memória secundária e atribui a memória libertada a um novo processo
    => comunicação entre o gestor de memória e o gestor de processos
  - o contexto do utilizador é copiado para memória secundária - código, dados e *stack*
  - associação de endereços é simples se feita em tempo de execução
  - também permite balancear a carga do sistema
  - gestão da área de *swap* em memória secundária

Gestão de Memória

- *Swapping*



Gestão de Memória

- *Swapping* (cont.)
  - melhora em média o tempo de resposta do sistema, no entanto
  - custo de *Swapping* = *2R*
    - S - unidades de memória ocupada
      D - unidades de memória por bloco de memória secundária
      R = S/D - número de *writes/reads* para copiar um processo de/para memória secundária
    - custo para o processo é o tempo que o processo gastará competindo para reaver a memória

Gestão de Memória

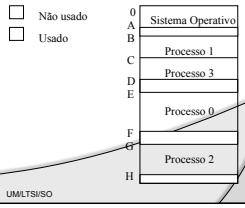
- *Swapping* (cont.)
  - *thresholds*
    - **S x T** - desperdício de recursos para um processo que gasta S unidades de memória quando está bloqueado T unidades de tempo.
      - Para o *swapping* ser efectivo, S tem de ser grande (de modo a ser atribuído a outro processo) e T >> 2R.
    - **S x T´** - mede a utilização da memória para um processo há T´ tempo em memória
  - *swapping* e I/O

Gestão de Memória

- Técnicas para atribuição de memória
  - Contígua
    - Partições fixas ou Partições variáveis
  - Não Contígua
    - Paginação ou Segmentação
- Principais questões
  - Protecção entre partições
  - Fragmentação
    - fragmentação interna
    - fragmentação externa
  - Desempenho

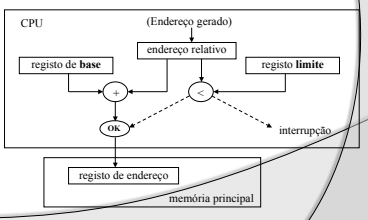
Gestão de Memória

- Atribuição contígua de memória
  - A memória principal é dividida em partições:
    - o sistema operativo é um programa residente, normalmente mantido nos endereços mais baixos da memória, junto ao vector de interrupções.
    - Processos do utilizador mantidos na restante memória.



Gestão de Memória

- Atribuição contígua de memória (cont.)
  - Protecção
    - os endereços gerados são comparados com o registo de limite para garantir que endereços externos à partição não sejam acedidos



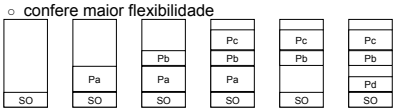
Gestão de Memória

- Atribuição contígua de memória (cont.)
  - Método das partições fixas
    - solução mais simples é dividir a memória em partições pré-definidas (iguais ou distintas)

    Questões:
    - fragmentação interna
      - perda de memória devido à alocação de espaço para processos cujos espaços de endereçamento são menores que a partição

Gestão de Memória

- Atribuição contígua de memória (cont.)
  - Método das partições de tamanho variável
    - a memória é atribuída a um processo de acordo com o espaço requerido
    - elimina a fragmentação interna
    - confere maior flexibilidade



  - introduz o problema da fragmentação externa
    - pequenos espaços de memória não podem ser atribuídos

Gestão de Memória

- Atribuição contígua de memória (cont.)
  - Método das partições de tamanho variável (cont.)

    Questões:
    - como gerir o espaço livre?
      - Qual o bloco de memória a atribuir?
      - Quais os blocos que podem ser fundidos?

      => mecanismo de gestão: o SO mantém o conjunto de partições ocupadas e livres

Gestão de Memória

- Atribuição contígua de memória (cont.)
  - Método das partições de tamanho variável
    - Fragmentação
      - externa – existe memória livre suficiente para satisfazer um pedido, mas não é contígua. Pode corresponder a 1/3 de memória existente!
      - interna – em alguns casos, a memória atribuída pode ser ligeiramente superior à pedida; a diferença é memória interna à partição, mas não usada.
      - Compactar a memória livre reduz a fragmentação externa
        - Só podemos compactar a memória se a associação de endereços é dinâmica e feita em tempo de execução.

Gestão de Memória

- Algoritmos genéricos para atribuição dinâmica de espaço
  - *First fit:* atribui o primeiro bloco livre que é suficientemente grande para satisfazer o pedido de memória
  - *Next fit:* pequena modificação ao *First fit.* Atribui o primeiro bloco livre que é suficientemente grande, mas começa a procura a partir do último ponto
  - *Best fit:* procura a lista completa e escolhe o menor bloco livre que é suficientemente grande para satisfazer o pedido de memória
  - *Worst fit:* procura a lista completa e escolhe o maior bloco livre

Gestão de Memória

- Atribuição não contígua de memória
  - **Paginação**
  - Segmentação

Gestão de Memória

- Paginação
  - o espaço de endereçamento lógico de um programa pode ser não contíguo; sempre que existe espaço livre, este é atribuído a um processo que o requisita.
  - A memória física é dividida em blocos de um só tamanho chamados *frames* (o tamanho é uma potência de 2, normalmente entre 512 *bytes* e 8192 *bytes*).
  - O espaço de endereçamento lógico é dividido em blocos do mesmo tamanho chamados páginas.
  - O gestor de memória mantém a lista de *frames* livres.
  - Para executar um programa de tamanho `n` páginas, é necessário encontrar `n` *frames* livres e carregar o programa.

Gestão de memória

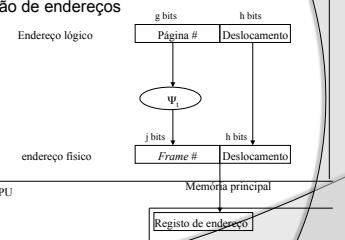
- CPU gera endereços em que uma parte identifica a página e a outra que identifica o endereço dentro dessa página (*page offset*)
- Do ponto de vista do programa o espaço de endereçamento lógico é linear - só tem uma componente
- Suporte de hardware permite transformar/ substituir parte do endereço correspondente ao número de página pelo respectivo número de *frame*.

Gestão de Memória

- Paginação
  - Esquema de tradução de endereços

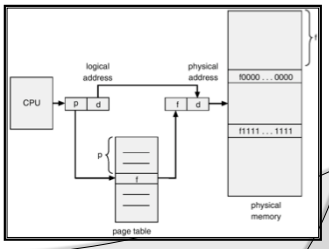
Os endereços gerados pelo CPU são divididos em:

• *Número de página (p)* – utilizado como argumento da função de tradução de endereços que retorna o endereço base da *frame* em memória física que contém a página.
• *Deslocamento (d)* – combinado com o endereço base, define o endereço físico final.

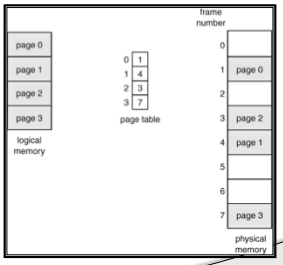


Gestão de Memória

- Paginação
  - Tabela de páginas



Gestão de Memória



Gestão de memória

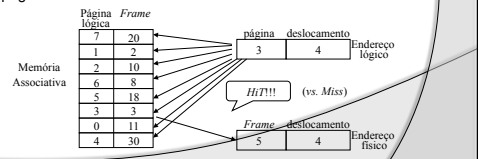
- Com paginação não há fragmentação externa
- Mas existe fragmentação interna
- Se o tamanho dos processos for independente do tamanho de página e se o tamanho de página for 4K qual é que seria em média a fragmentação interna por processo?
  - Pior caso: tamanho do processo = n*4K+1
  - Melhor caso: tamanho do processo = n*4K
  - Caso médio: 50% do tamanho de página (2K por processo)
- Que implicações é que isso têm no tamanho ideal das páginas?
  - Páginas mais pequenas diminuem fragmentação interna
  - Páginas mais pequenas aumentam *overhead*

Gestão de Memória

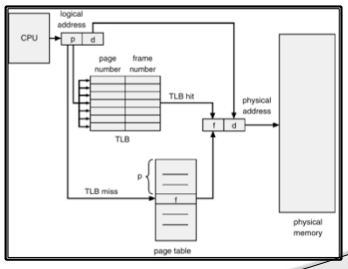
- Paginação
  - Implementação da tabela de páginas
    - A tabela de páginas global é mantida em memória principal
    - O registo *Page-table base register (*PTBR) aponta para o início da tabela de páginas
    - O registo *Page-table length register* (PRLR) indica o tamanho da tabela de páginas.
    - Neste esquema, cada acesso a dados/instruções requer dois acessos à memória: um para a tabela de páginas e outro para os dados/instrução.
    - Este problema pode ser resolvido pela utilização de uma cache implementada por hardware de procura rápida, chamada memória associativa ou *translation look-aside buffers (TLBs)*

Gestão de Memória

- Paginação - Implementação da tabela de páginas
  - memória associativa de alta velocidade
    - baseia-se na observação de que a maior parte dos programas tende a referenciar várias vezes um número pequeno de páginas
    - o acesso é feito em paralelo
    - se a memória não contém a página em questão (*Miss*), a tabela de páginas é consultada e criada uma nova entrada



Gestão de memória



Gestão de Memória

- Paginação - Protecção
  - acessos a memória não atribuída a um processo são facilmente detectados através do limite da tabela de páginas
  - acessos a páginas que não fazem parte do espaço endereçamento lógico de um processo é controlado através dos *valid/invalid bit* na entrada da tabela de páginas
  - permissões de acesso são associados a cada página

Gestão de Memória

- Paginação - Partilha
  - Código partilhado
    - Uma cópia de código *read-only* partilhado por vários processos (i.e., editores de texto, compiladores, sistemas de janelas).
    - As páginas partilhadas fazem parte do espaço de endereçamento dos vários processos
    - As páginas partilhadas têm a mesma localização no espaço lógico de todos os processos.
  - Dados e código privado
    - Cada processo mantêm os dados e código privados em páginas separadas.
    - As páginas privadas não têm restrições de localização no espaço de endereçamento lógico.

Gestão de Memória