Programação de Sistemas

Paginação de memória

Introdução (1)

A. Ao ser lançado um processo, o código e os dados são colocados pelo SO em posições arbitrárias.
   Logo, um compilador **não pode assumir** endereços fixos para o código e para os dados!
B. Com o aumento da complexidade dos programas e do número dos processos em execução, a memória RAM disponível pode não ser suficiente para todas as necessidades de cada processo.

- A paginação constitui uma solução eficiente para os dois problemas .

Introdução (2)

- Os compiladores e editores de ligação manipulam endereços virtuais (ou lógicos).
- Num processo em execução, o SO transcreve os endereços virtuais para os endereços reais (ou físicos).

Nota: A memória virtual foi implementada pela primeira vez no computador Atlas, na Universidade de Manchester/UK.

O primeiro microprocessador da Intel a disponibilizar memória virtual, em modo protegido, foi o 80286 (processador de 16 bits).

Introdução (3)

[Definição] **MMU**-"Memory Management Unit" é a unidade que translada o endereço virtual (ou lógico) para o endereço efectivo (ou real).

Ex: se o endereço virtual for 0x32f e o endereço base for 0x14100, o endereço efectivo é igual a 0x1442f.

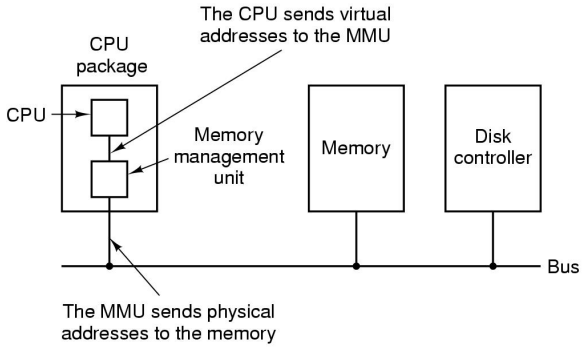


Figura 4-9, Modern Operating Systems

Paginação (1)

- A memória central é dividida por zonas de pequena dimensão, designadas por **molduras** (“frames”) ou páginas físicas, todas do mesmo tamanho.
- O espaço de endereçamento do processo é também dividido por zonas de pequena dimensão, designadas por **páginas** (“pages”) ou páginas virtuais, com o mesmo tamanho das molduras.
- Para cada processo, o SO mantém na memória central uma **tabela de páginas**. Para cada página virtual, a tabela de páginas indica[1] a moldura onde a página está carregada

[1] com informação extra, a ver mais à frente.

Paginação (2)

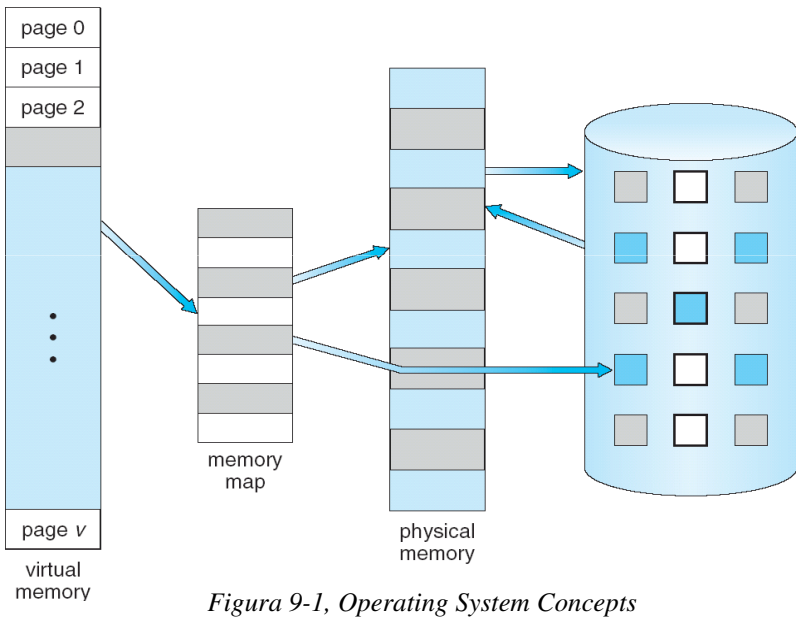


Figura 9-1, Operating System Concepts

Paginação (3)

- O endereço virtual contém duas partes
  - Prefixo – número de página, que indexa uma tabela de páginas
  - Deslocamento (“offset”)

Pag #	offset

- O endereço físico é determinado nos seguintes passos:
  1. Recolher número de página.
  2. Usar o número de página para indexar a tabela de página, e recolher o endereço base da moldura.
  3. O endereço real é obtido somando o endereço base da moldura com o deslocamento.

Paginação (4)

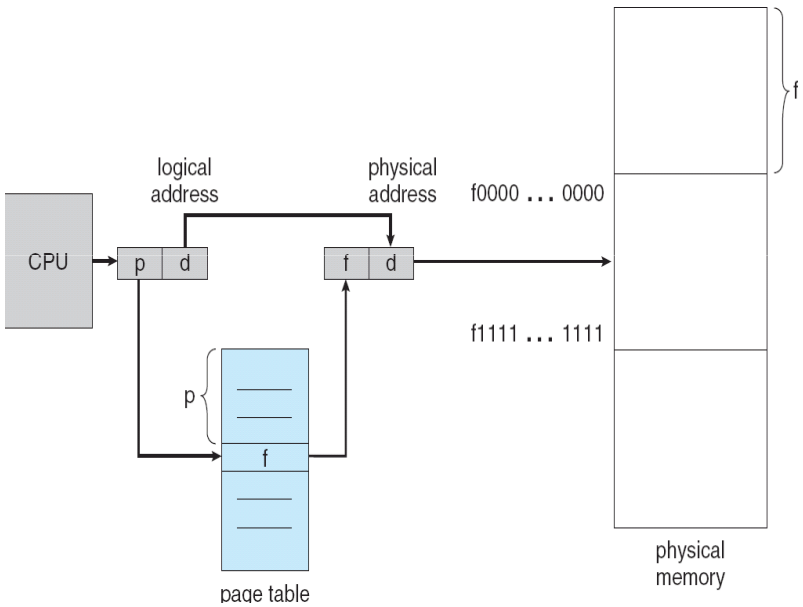


Figura 8-12, Operating System Concepts

Paginação (5)

Exemplo1: endereço virtual 0x2004, com páginas de 4KB (12 bits).
A entrada 0x2 refere a moldura 0x6. O endereço virtual é transcrito para o endereço real 0x6004.

Nota: neste exemplo, VPN possui 4 bits e PPN possui 3 bits

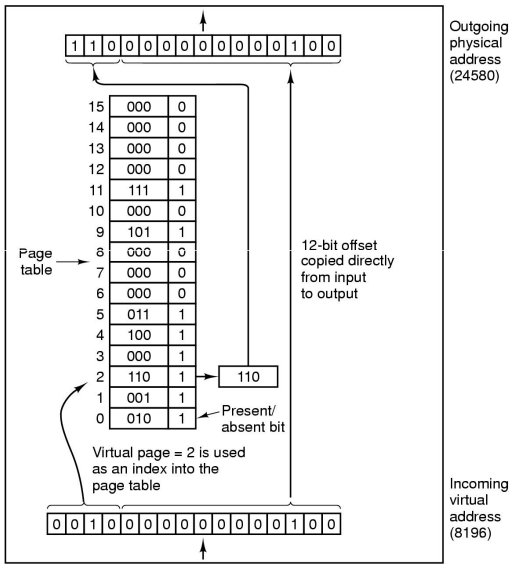
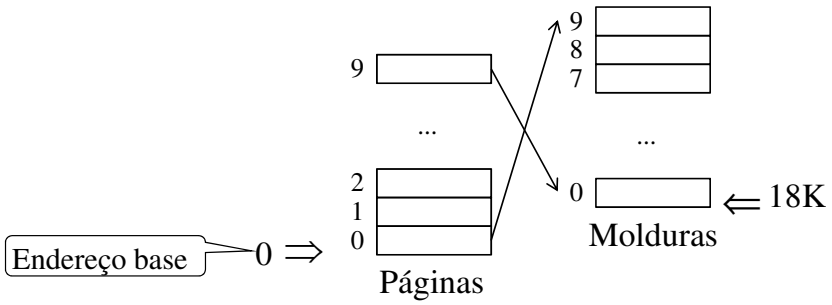


Figura 4-11, Modern Operating Systems

Paginação (6)

Exercício: Considere um sistema VM com páginas de 2KB e um processo de 10 páginas.
O processo reside na memória real em páginas de endereços sucessivos por ordem inversa (i.e., a primeira página é colocada na última moldura, a segunda página é colocada na penúltima moldura e assim sucessivamente) a partir de 18KB.
Qual a localização física do endereço virtual `6325` (decimal)?



Paginação (7)

1. Determinar a página e o deslocamento (“offset”).
   Fácil, basta efectuar uma divisão!

   6325 | 2048
   181 | 3

   Deslocamento — Página

2. Determinar o número da moldura.
   Fácil, uma vez que as molduras residem na memória real por ordem inversa!

   Índice = (10-1)–número página = 9-3 = 6

3. Calcular endereço real.

   Endereço base da moldura

   ER = Endereço base + índice*dimensão + deslocamento

   = 18K+6*2K+181 = 18432+12288+181 = 30901

Formato da tabela (1)

- As entradas na tabela (PTE-“Page Table Entry”) possuem diversos campos:

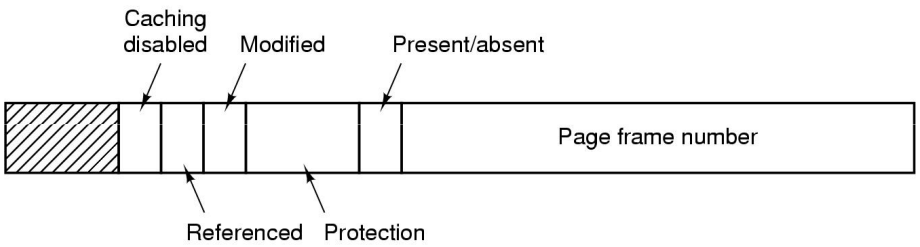


Figura 4-13, Modern Operating Systems

  - Endereço da moldura (Frame number).
  - Inibição “caching”: impede a página residir na cache, por exemplo quando é mapeada para dispositivo E/S.

Formato da tabela (2)

- Bandeiras
  - Referenced : a página foi solicitada pelo processo, quer para leitura quer para escrita (se ainda não foi é candidata a ser substituída).
    Nota: bandeira útil para alguns algoritmos de substituição.
  - Modified (ou Dirty): a página foi modificada e ainda não foi salvaguardada para disco.
  - Validity: indica se página reside, ou não, na RAM.
- Idade da página (Nota: útil para algoritmos de substituição)
- Protecção: permissão para escrita, página de utilizador ou núcleo,…

- A tabela de páginas reside em memória e é acedida a partir de dois registos do µP
  - PTBR ("Page Table Base Register"), para a localização em memória da tabela.
    Nota: no Pentium o PTBR é o registo CR3.
  - PTLR ("Page Table Length Register"), para a dimensão da tabela.

Formato da tabela (3)

- A dimensão da moldura, obrigatoriamente potência de 2, é determinada em diversas formas:
  1. Fixa, determinada na fase de desenho do µP: opção dos sistemas VM iniciais (ex: Intel 80286).
  2. Estática: em cada instante, o VM só admite 1 dimensão
     - Intel Pentium: 4KB-12 bits ou 4MB-22 bits.
       Nota 1: no Linux, os processos utilizador usam páginas de 4KB e o núcleo usa páginas de 4MB-"jumbo pages".
       Nota 2: no Linux, a dimensão da moldura é indicada no ficheiro `asm/page.h`.
     - Motorola MC68030: valor entre 256B e 32KB
  3. Dinamicamente configurável: molduras de dimensões distintas podem coexistir na VM, desde que alinhadas
     - UltraSPARC: 8KB, 64KB, 512KB e 4MB

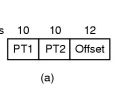
Redução do espaço de tabelas (1)

- A dimensão das molduras é problema de desenho:
  - Desperdício médio devido a fragmentação interna é ½ página por processo, logo o ideal é ter molduras pequenas.
  - Mas, molduras pequenas aumentam dimensão da tabela de páginas para cada processo, o que gasta memória!
  - Tipicamente, um computador reserva 4KB-64KB para as tabelas.
- O espaço de memória ocupado pela tabela de páginas é dado por dim * $2^{N_{VP}}$
  - dim é o espaço da PTE
  - N_{VP} é o número de bits do número de página virtual.

Exemplo: se dim=4B e VPN=20, a tabela de páginas de um processo ocupa 4MB. Se estiverem a correr 100 processos, 400MB de RAM são ocupados só para tabelas.
Ouch ☹!

Redução do espaço de tabelas (2)

- Espaço de tabelas por reduzido por tabelas multinível, tabelas dispersas ("hashed") e tabelas invertidas.
- Aqui abordamos apenas as tabelas multinível.
- Os bits da página virtual são divididos em várias partes.
  - Os bits mais elevados indexam a tabela do 1º nível (usualmente designada por **directório**).
  - Cada entrada do directório referencia a tabela do 2ºnível (tabela de páginas), indexados pelos bits mais baixos.



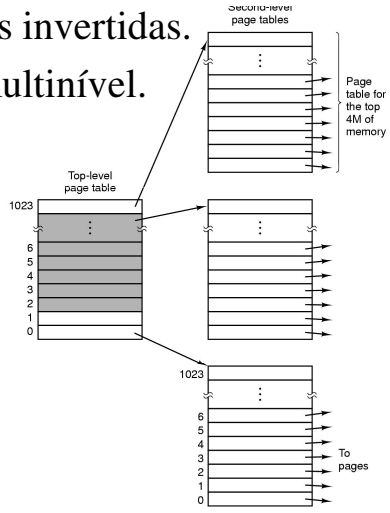


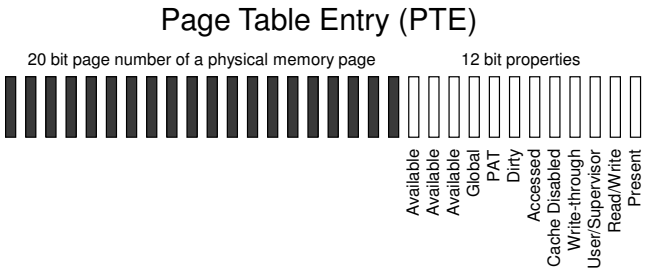
Figura 4-12, Modern Operating Systems

Redução do espaço de tabelas (3)

- A estruturação das tabelas por vários níveis diminui espaço ocupado pelas tabelas em duas formas:
  a) Se o 1º nível indicar um endereço inválido, então não é necessário alocar os espaços das tabelas do 2º nível.
  b) É muito vulgar os endereços possuirem as áreas de código, pilha e dados muito densas e grandemente separadas. Os processos podem assim ter apenas 3 entradas na tabela do 2º nível, que não necessitam de ser contíguas.
- Cada µP implementa determinado número de níveis
  - Intel 386,486 : 2 níveis fixos
  - Intel Pentium : 1 ou 2 níveis
  - Sun SPARC, Intel Core : 3 níveis
  - Motorola 680x0 : suporta até 4 níveis.

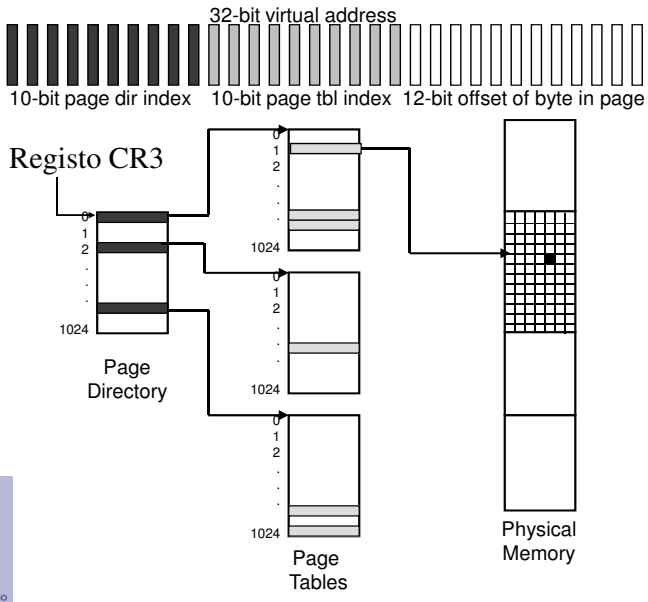
Redução do espaço de tabelas (4)

- A Intel disponibilizou VM partir do 386.
- A PTE do 386,486 e Pentium tem o seguinte formato



- No Pentium, o número de tabelas depende do tamanho da moldura (fixada por um bit do registo CR4)
  - Moldura de 4MB: directório
  - Moldura de 4KB: directório e página.

Redução do espaço de tabelas (5)

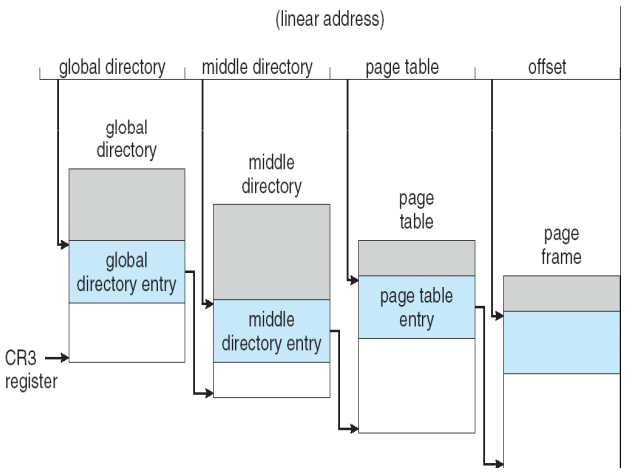


- 10 bits de topo indexam página de directório, cujo valor aponta para a tabela de página.
- 10 bits intermédios indexam tabela de página, cujo valor aponta para a moldura.
- 12 bits de base são deslocamento para o Byte na página física.

- Em cada passa são exercidas verificações para garantir que a página desejada se encontra disponível na memória e que o processo possui direitos a aceder à página.

Redução do espaço de tabelas (6)

- Nos µP de 64 bits, por exemplo o Intel Core 2, o directório do Linux é dividido em 2: global e intermédio.



Falha de página (1)

- Inicialmente, todas as páginas encontram-se ausentes com V=0.

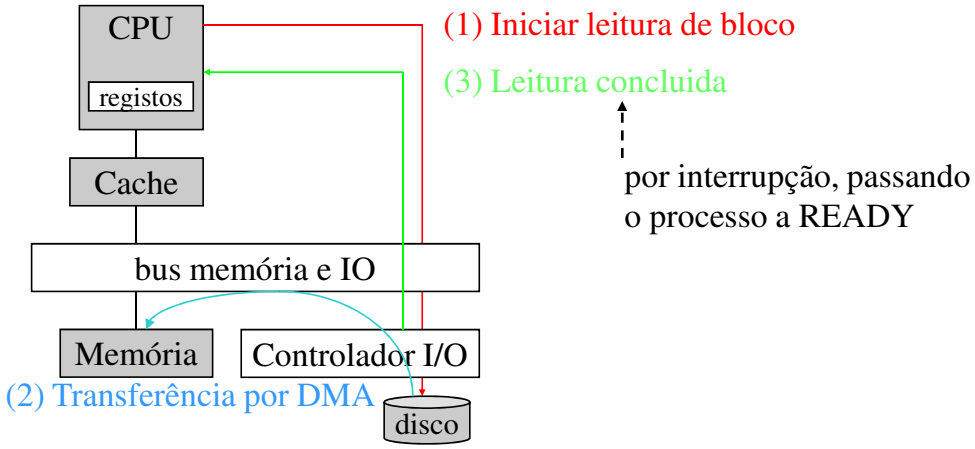
1. Quando o programa acede a um endereço virtual (código, dados ou pilha), verifica V na entrada identificada pelo número de página virtual (prefixo do endereço virtual).
   - Se V=1 (acerto-"hit"), a página está presente na memória central e o número de página é substituído pela moldura presente na PTE.
   - Se V=0 (falha-"miss"), ocorre uma excepção PF ("page fault") **falha de página**.

Falha de página (2)

A ocorrência de PF tem dois resultados:

1. O endereço é inválido (cai fora dos limites do processo, tentativa de escrita em página "read-only") e o programo é abortado.
2. O endereço é válido e carrega a página. O carregamento de página em falha P segue os passos:
   i. Identifica uma moldura vazia F (se necessário liberta uma moldura ocupada)
   ii. Carrega a página do disco para a moldura F.
   iii.
```
TPE[P].frame = F;   // localização da página
TPE[P].V = 1;       // página reside na RAM
TPE[P].M = 0;       // página ainda não foi modificada
TPE[P].age = 0;     // idade zero (para algoritmo LRU)
```
   iv. Fixados os valores de protecção apropriados (ex: página de código é "read-only")
   v. Retoma a instrução que provocou excepção PF.

Falha de página (3)



- A falha de página demora milhões de ciclos de relógio a processar (tipicamente vários ms, enquando acesso à memória central ronda a décima parte de µs).

Falha de página (4)

- Tempo de acesso efectivo EAT ("Effective Access Time")
  - Caso a página resida na memória, existem dois acessos (um à tabela, outro ao dado).
  - Em caso de falha, é necessário adicionar o tempo de salvaguarda da moldura (se a página tiver sido modificada) e o tempo de carregamento da página em falta.

$$EAT = (1 - p) * (t_{acesso\_tab} + t_{acesso\_mem}) + p\,(t_{acesso\_tab} + (1\text{-}m)*t_{down} + m*(t_{up} + t_{down}))$$

p : taxa de falha
m : taxa de modificação da página

Desprezado por ser muito menor

- Estratégias para minimizar EAT
  A. Diminuição do tempo de acesso à tabela t_{acesso_tab} (arquitectura de HW)
  B. Diminuição da taxa de falha p (algoritmo de substituição páginas)

Falha de página (5)

Problema: Qual é o EAT no seguinte caso?

- A taxa de falha de página é 1 em cada 100 mil acessos.
- Acesso à tabela é dado por t_{acesso_tab}=8 ns e acesso à memória é dado por t_{acesso_mem}=100 ns.
- O carregamento de uma página é feito em 8 ms, a salvaguarda de uma página modificada é feita em 20 ms.
- A taxa de modificação de páginas é de 10%

$$EAT = (1-10^{-5})*108 + 10^{-5}(0.9*8x10^6+0.1*(2x10^7+8x10^6))$$
$$\cong 108 + 10^{-5}(7.2x10^6+2.8x10^6)$$
$$\cong 210 \text{ ns}$$

Nota: bastava p aumentar para 1 por 10 mil, para EAT passar para 1.1 µs (p=1/2000, EAT=5.1 µs)

Permuta de páginas (1)

- Quando ocorre uma excepção PF, é necessário carregar do disco a página em falta.
  - Se existir uma moldura livre, usá-la.
  - Se não existir uma moldura livre, usar um algoritmo de substituição de páginas para seleccionar uma moldura ocupada (vítima).

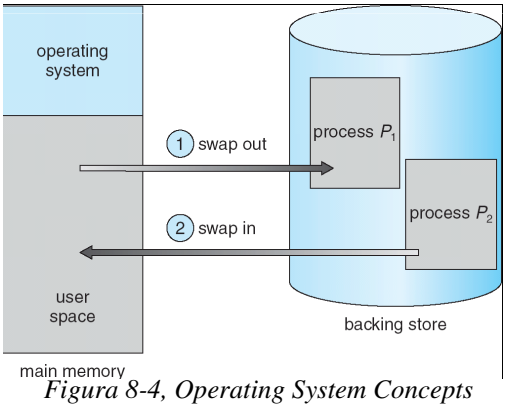


Figura 8-4, Operating System Concepts

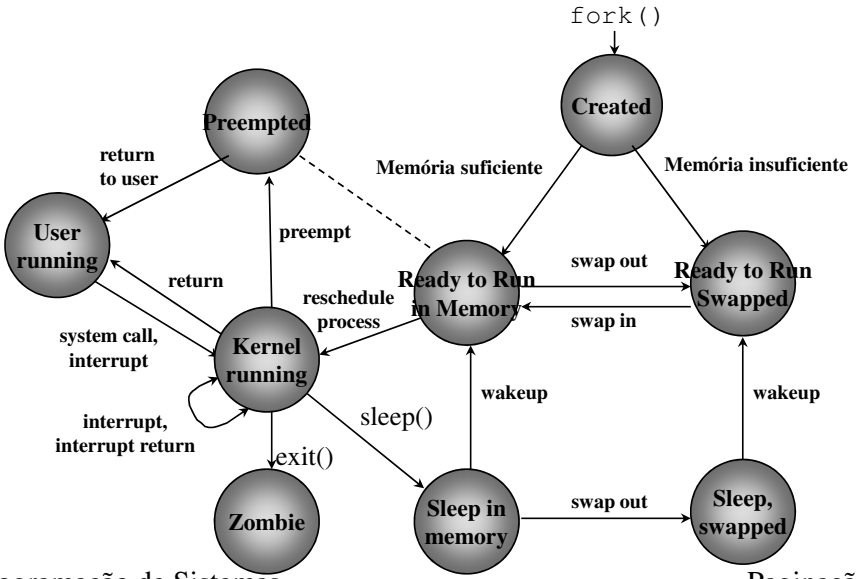
Permuta de páginas (2)

[Definição] A **permuta** de página ("swap") é a operação de envio para disco de uma página vítima para libertar espaço a fim de carregar uma página em falta.

- O Linux exige uma partição de disco para guarda temporária de páginas alteradas (ex: páginas de variáveis), que são trocadas.
  - Tipo de ficheiro da partição de swap é 0x82 (Linux swap)
  - O espaço da partição de swap deve ser, no mínimo, a de memória RAM.
    Normalmente a partição possui o dobro da RAM, devido aos espaços de trabalho ("working set")
- No Linux, o espaço de memória virtual é dado pela soma da memória central com a partição swap.

Permuta de páginas (3)

Diagrama completo de transições de estados de um processo

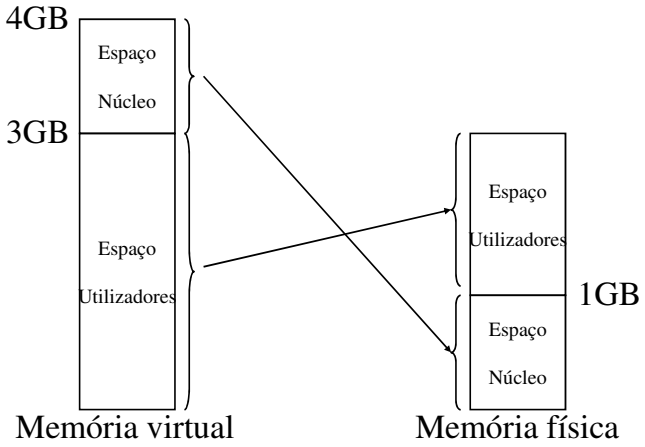


Algoritmos de substituição

- O algoritmo de substituição de páginas deve minimizar o número de substituições. O número de carregamento de páginas é limitado entre
  - Mínimo: número de páginas distintas
  - Máximo: total de pedidos de acesso
- Para a mesma sequência de referências, as substituições devem diminuir com o aumento de memória central – condição de Belady.
- Algoritmos mais divulgados:
  - FIFO (First In,First Out)
  - OPT
  - LRU (Least Recently Used)
  - WS (Working Set)

Divisão de memória no Linux 32 (1)

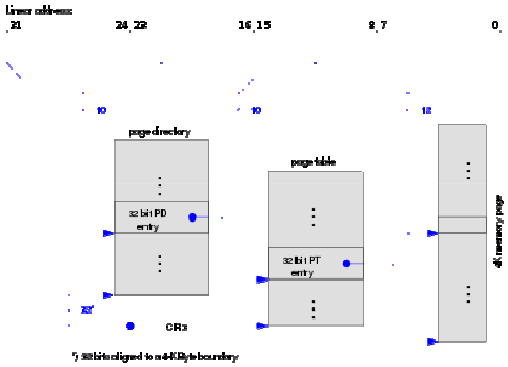
- Os processadores de 32-bit endereçam 4GB.
- No Linux, as memórias (virtual e física) são divididas em duas partes:



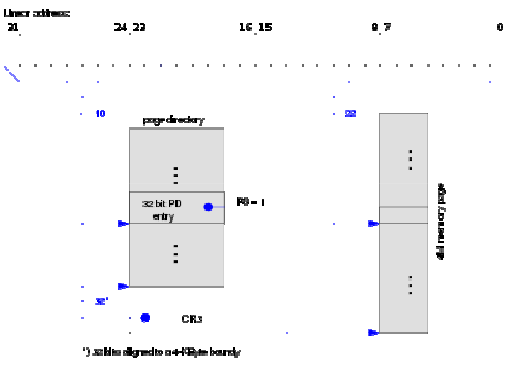
Divisão de memória no Linux 32 (2)

- Para aumentar a memória física para além dos 4GB, os processadores Pentium incluem o *PAE-Physical Address Extension*, alargando bus de endereços de 32 para 36 bits:
  - Memória física aumentada para 36GB.
  - Para cada processo, a memória virtual continua nos 4GB (1GB reservado para espaço de núcleo).
    O MMX mapeia cada 4GB do processo para 64GB.
  - PAE em efeito através do bit 5 do registo CR4.
    Cada entrada das páginas de directório e de tabela passam a ter 64 bits.

Divisão de memória no Linux 32 (3)

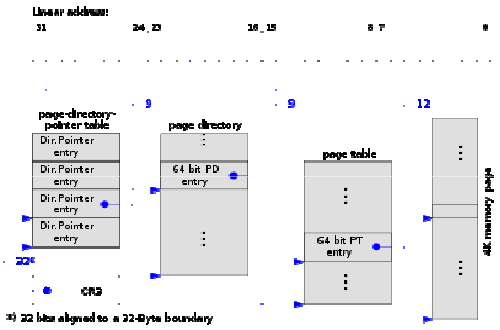


Moldura 4KB, sem PAE

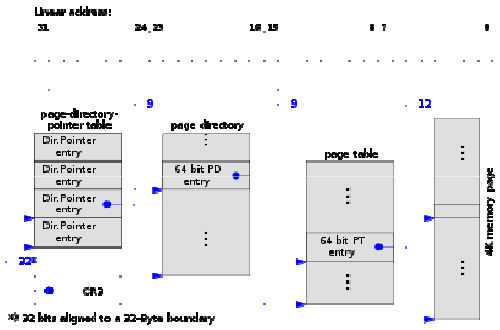


Moldura 4MB, sem PAE

Divisão de memória no Linux 32 (4)



Moldura 4KB, com PAE



Moldura 4MB, com PAE

Alterar espaço swap no Linux (1)

- O espaço swap deve ser duplo da memória RAM.
- Quando se instala mais RAM tornar-se necessário incrementar o espaço de swap em disco.
  Altera o espaço de swap em disco é feito com privilégio de root

A. Para determinar o espaço swap actual, consultar `/proc/swaps`

```
[rgc@asterix Integration]$ more /proc/swaps
Filename                        Type            Size    Used    Priority
/dev/mapper/VolGroup00-LogVol01 partition       2031608 88480   -1
```

Alterar espaço swap no Linux (2)

B. Criar ficheiros
   - Para melhorar desempenho, criar mais de 1 ficheiro swap.
   - Ficheiros swap devem ser contínuos, criados por `dd`.

```
$ cd /var/tmp
$ dd if=/dev/zero of=swapfile1 bs=1024 count=1048576
```

fonte dimensão

C. Transformar ficheiro em área Swap

```
$ /sbin/mkswap -c -v1 /var/tmp/swapfile1
```

D. Autorizar swap

```
$ /sbin/swapon /var/tmp/swapfile1
```


Criação de processos e VM

- A memória virtual permite mais rapidez no `fork()`, através da técnica COW-"Copy-On-Write"
  - Processos pai e filho partilham as mesmas páginas
  - Se um processo alterar uma página, só então esta é copiada.