NOVA FACULDADE DE CIÊNCIAS E TECNOLOGIA DEPARTAMENTO DE INFORMÁTICA

Fundamentos de Sistemas de Operação

LEI - 2023/2024

Vitor Duarte
Mª. Cecília Gomes

1

Aula 12

- Gestão de memória, atribuição de espaço livre
- Paginação a pedido

- OSTEP: cap. 14, 17-17.3, revisão AC: cap. 15, 18, 19

2

Níveis de gestão de memória

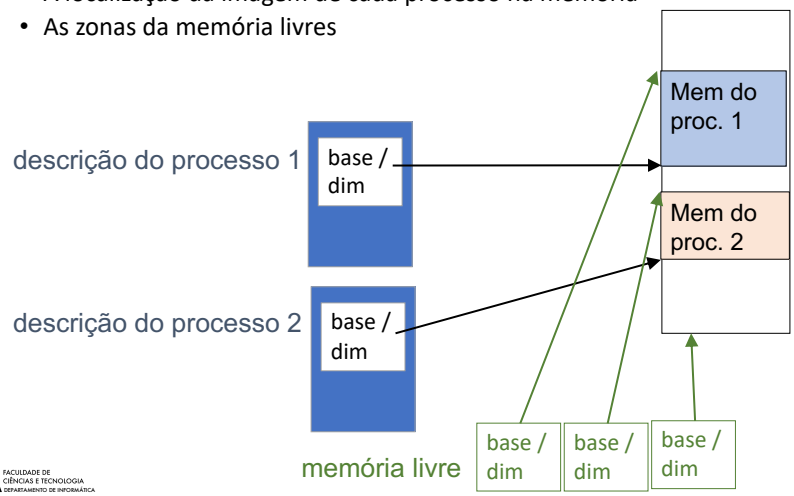
- SO
  - Gere a memória física existente, atribuindo-a aos processos e a si próprio
  - Gere o seu próprio espaço para os seus dados, código e pilha
  - Controla os espaços virtuais de cada processo, suas proteções e mapeando destes em memória real
  - Configura o MMU a cada troca de contexto:
    - A MMU resolve os endereços durante a execução das instruções
    - As instruções de alteração da MMU são priveligiadas
    - No estado do processo está a sua configuração (registos base e limite de segmentos, ou registos base e limite da tabela de páginas)
  - *Gere a memória secundária (swap/paging) para oferecer memória virtual -> mais à frente*

Níveis de gestão de memória

- Processo
  - Cada processo/programa gere o seu espaço virtual
  - Na compilação/ligação define dimensões dos segmentos lógicos para diferentes fins e suas permissões:
    - Código, Pilha, Dados estáticos, Dados dinâmicos (heap)
  - Pede ao sistema o mapeamento do espaço virtual
    - No início e durante o funcionamento
  - A zona de heap atribuída pelo SO é gerida pelo processo/programador:
    - normalmente usando o runtime da linguagem ou bibliotecas standard: malloc/free; new/delete; new/GC; ...
  - *Outras zonas, explicitamente pelo programador:*
    - *mmap/munmap; ... → mais à frente*

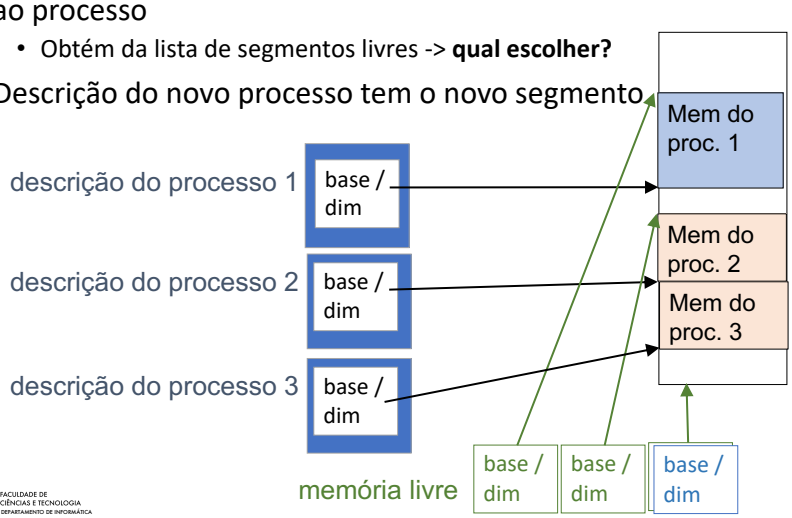
Gestão por segmentos contíguos

- O kernel do SO mantém:
  - A localização da imagem de cada processo na memória
  - As zonas da memória livres



Novo processo ou troca de imagem

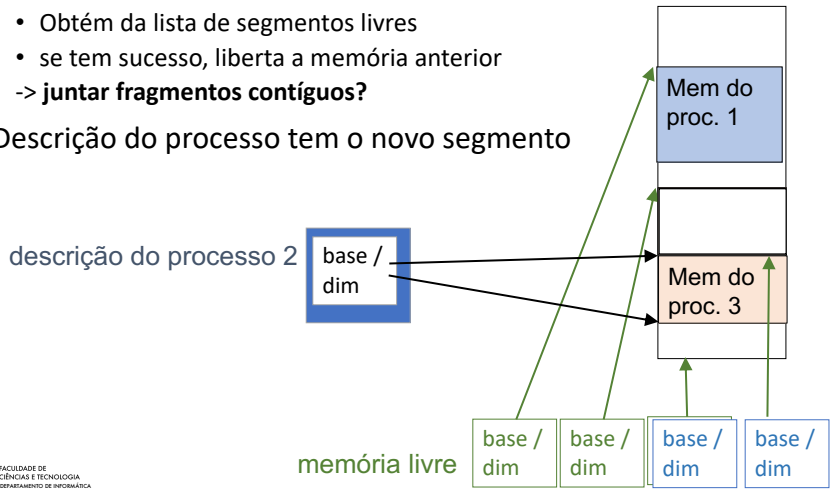
- Novo processo (fork): o kernel do SO atribui memória livre ao processo
  - Obtém da lista de segmentos livres -> **qual escolher?**
- Descrição do novo processo tem o novo segmento



Novo processo ou troca de imagem

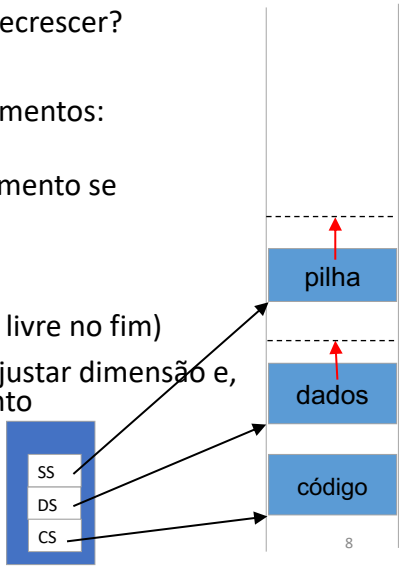
- Troca de imagem (exec): O kernel do SO atribui memória livre ao processo
  - Obtém da lista de segmentos livres
  - se tem sucesso, liberta a memória anterior

  -> **juntar fragmentos contíguos?**
- Descrição do processo tem o novo segmento



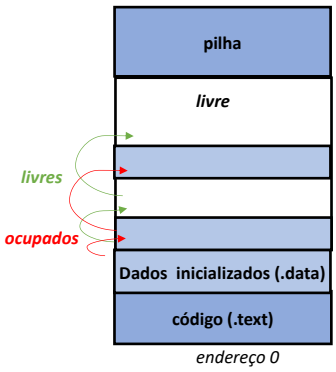
Segmentos dinâmicos

- Os processos podem crescer? E decrescer?
- Numa arquitetura com vários segmentos:
  - código, dados, pilha, …
- O SO pode fazer crescer cada segmento se
  - há memória física disponível
  - há memória física livre à frente
  - ou movendo o segmento
- O SO pode reduzir segmentos (se livre no fim)
- Alterações ao processo: apenas ajustar dimensão e, eventualmente a base do segmento



Gestão interna ao processo

- Gestão do seu Heap, a cargo do processo
  - em bibliotecas ou runtime da linguagem
  - exemplo: malloc/free, new/delete, etc...
- gerir memória atribuída em blocos livres e ocupados
  - pode precisar de pedir mais memória ao SO para o processo crescer
  - exemplo: malloc(size)
    - atribui dos blocos livres, eventualmente do último no fim do heap, ou pede mais memória ao SO
- Também sofre de fragmentação
  - mas não pode compactar...



Estratégias para gestão de espaço

- As estratégias devem ser bastante eficientes em espaço e tempo
- Escolher de entre os espaços livres:
  - estrategias para blocos de tamanho variável: best-fit, worst-fit, first-fit, etc
  - atribuir tudo ou partir o bloco (**spliting**)
- Libertar espaço
  - Juntar aos espaços livres, com eventual fusão com os existentes (**coalescing**)
- Combatendo a fragmentação
  - Mover blocos ocupados para compactar e juntar os blocos livres: possível pelo SO pois os endereços virtuais da arquitetura permitem mover os processos; dificil dentro do processo

Gestão de memória pelo SO

- Endereçamento virtual por segmentos ou páginas permite multiprocessamento com proteção e isolamento
- Hardware de endereçamento (MMU) mais complicado e lento com páginas
  - mas usa a TLB para reduzir o problema
- Páginas não necessitam de estratégias de atribuição complicadas
  - Sempre best-fit
- Uso de páginas é mais flexível e poderoso
  - deixa de ter fragmentação externa (reduz a interna) não necessita de compactar
  - é fácil crescer e reduzir os processos
  - permite partilha de frames (memória partilhada entre processo)
    - poupa memória, novas funcionalidades, …
  - e mais…

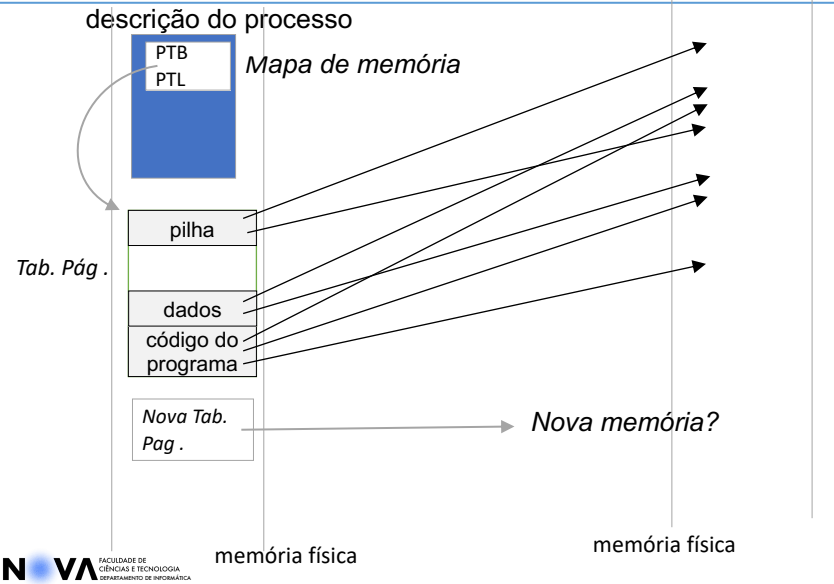
Gestão das páginas dos processos

- As páginas permitem maior eficiência na gestão da memória central e na criação de processos:
  - O novo processo começa com a mesma memória do pai
  - Ocupa menos espaço real
  - Na tabela, as páginas de **código** são marcadas como Exec, Read-Only e partilhadas
  - A copia da restante memória será "lazy": em ambas as tabelas, as áreas de dados e "stack" são marcadas como páginas Read-Only para COPY-on-WRITE

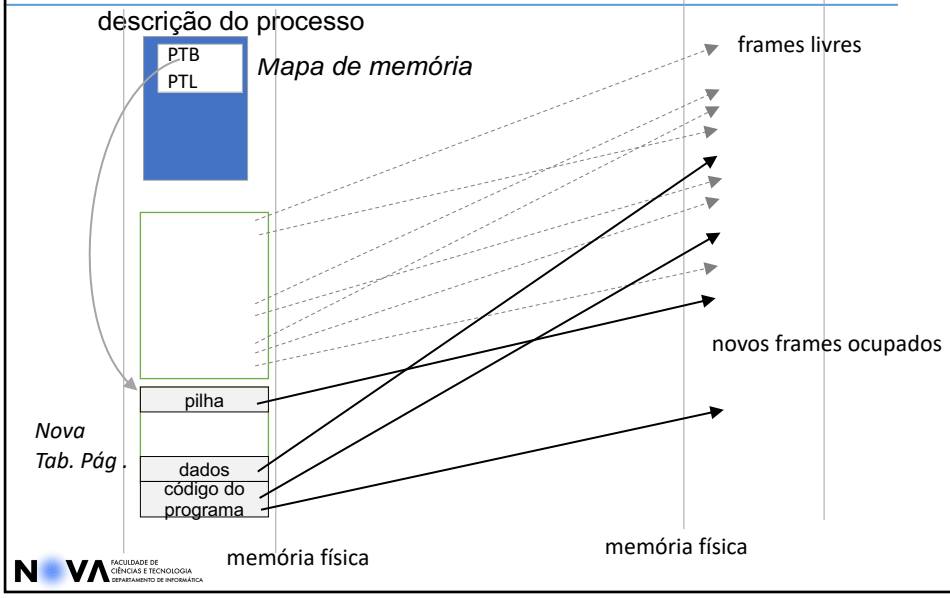
Copy-on-Write

- Só quando se tenta modificar a página partilhada é que se procede à sua duplicação (cópia)
  - Para a MMU a página não pode ser escrita
  - Quando se tenta um escrita (exemplo: mov para memória) é levantada uma interrupção pela MMU
  - O SO atende e atribui uma nova frame ao processo que toma o lugar da página que deu lugar à interrupção
    - copia conteúdo, se necessário
    - atualiza nas tabelas de páginas dos processos para leitura e escrita
- Esta técnica permite que o pai e o filho partilhem todas as páginas em memória (incluindo os dados e a pilha) que não sejam escritas.

No caso de nova imagem? exec*



No caso de nova imagem? exec*



15

Paginação a pedido (on-demand)

- O SO pode seguir também a abordagem "lazy" no carregamento da nova imagem do programa
- Carrega o novo programa por **paginação a pedido**:
  - Cada página virtual tem associada um bit de presença (poderá ser o de validade ou não)
  - só é mapeada em memória física (ou seja, só lhe é atribuído um *frame*) quando é referenciada
  - O *hardware* notifica a falta de página (**page fault**) por interrupção; passa ao SO o problema...
  - Se válida, o SO obtém um *frame* livre e inicia-o
    - por exemplo, com o conteúdo do respectivo ficheiro executável (dados ou código)

16

Atribuição de frames

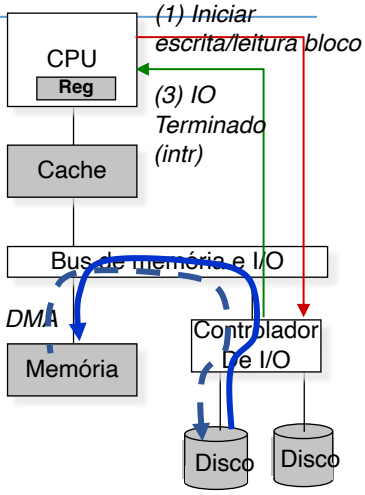
- O SO mantém numa tabela de frames se estão ocupadas/livres
- Quando é necessário, atribuir uma frame a uma página de um processo:
  - porque está em copy-on-write
  - porque é paginação a pedido
  - porque o processo pediu mais memória e pode pedir
- Obtém uma na tabela de livres
  - se encontra, marca ocupada, atualiza o seu conteúdo e a tabela do processo, que continua
- Se não? O que fazer?
  - guardar em disco uma frame ocupada e atribuir-la a esta página
  - semelhante às tecnicas das caches, mas agora entre RAM e disco
- **Memória Virtual**, por extensão da memória central para memória secundária

Tratamento de interrupções da MMU

- Acesso a um endereço numa página que não está em memória ou não é válida, é uma exceção e o SO intervém:
  - Se página inválida (fora da imagem atribuída ao processo) ⇒ aborta o programa (ex: SIGSEGV - "*Segmentation Fault*").
  - Se acesso inválido para as permissões ⇒ verifica se *copy-on-write*, se não aborta
  - Se válida mas falta a página ⇒ trata a falta de página
- Tratamento de falta de página (***page fault***):
  - Obtém uma "frame" livre. Se não houver, liberta uma usando uma política de substituição→ guarda-a em disco (swap out)
  - Carrega a página nesse "frame" (a zeros ou swap in)
  - Atualiza a tabela de páginas (nº frame e presente/válida)
- Retoma à instrução que provocou a interrupção

Tratando falta de página

- Para libertar uma *frame*, pode ter de a escrever para disco . . .
  - page out
- Para iniciar a nova *frame* pode ter de ler do disco . . .
  - page in
- O processo corrente pode ficar **bloqueado**
  - Outros processos podem executar até a falta estar resolvida
- SO desencadeia as transferências IO
  - Direct Memory Access (DMA)
- I/O Controller assinala o fim por interrupção
  - SO atualiza as respectivas tabelas de páginas
  - Processo volta a pronto (READY), execução retomada



Processo e sua memória

descrição do processo 1

Mapa de memória

Estado do CPU

Canais IO

Canais IO

núcleo do SO

Tab. Pag .

No início, e durante o funcionamento, podemos ter sempre uma imagem parcial em memória

pilha

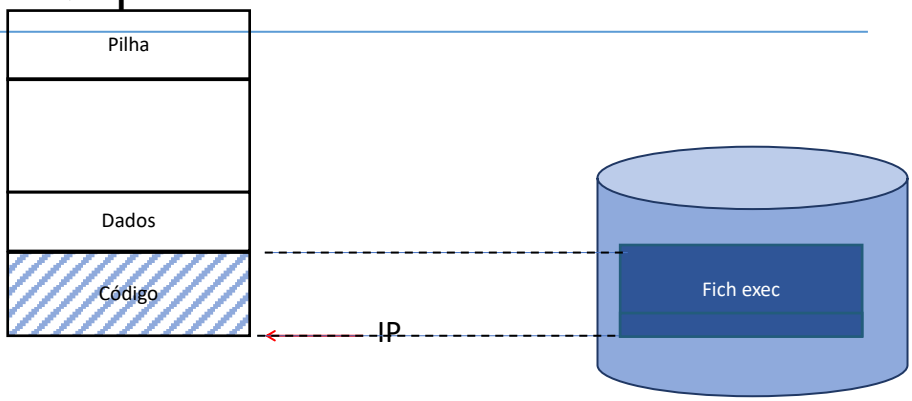
dados

código do programa

SO com Paginação a pedido

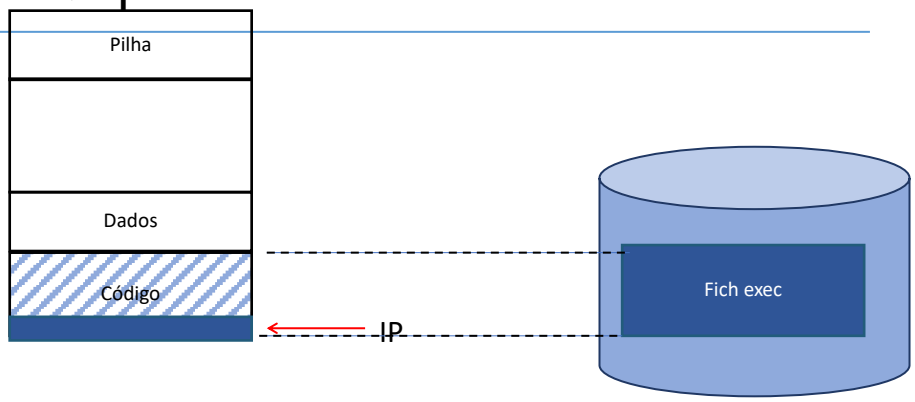
- O SO só mapeia a página para memória quando é referenciada (quando é pedida). Se o conteúdo já está num *frame* (ex: uma página de outro processo) tenta partilhar
  - Menos I/O (em principio)
  - Menos RAM utilizada
  - Tempo de resposta menor
  - Mais programas em RAM
- No caso do código, está sempre no ficheiro executável
  - É como se o ficheiro esteja mapeado em memória
  - A memória tem uma cópia do ficheiro, possivelmente parcial (como uma "cache" do ficheiro)
- Permite-se que o SO guarde *frames* em disco (*swap out*) quando tem falta de memória física
  - A paginação a pedido volta a carregar quando necessário

Mapeamento de ficheiro em memória



De início de um novo programa, a tabela só tem paginas marcadas como não presentes, mas baseadas no fich. executável.

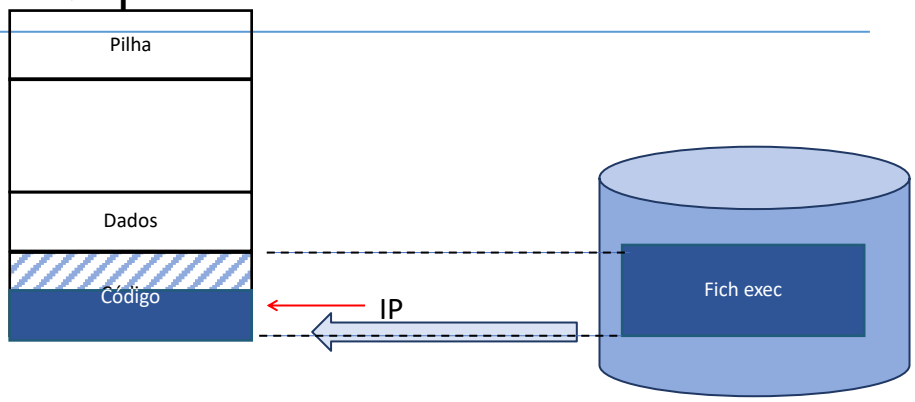
Mapeamento de ficheiro em memória



Parte do espaço do processo corresponde aos bytes no ficheiro. Vão sendo trazidos para memória a pedido ("on-demand")

23

Mapeamento de ficheiro em memória



Parte do espaço do processo corresponde aos bytes no ficheiro. Vão sendo trazidos para memória a pedido ("on-demand")

24

Memória virtual com paginação a pedido

- Permite **memória virtual** de dimensão arbitrária:
  - Usa o disco para "estender" a memória real
  - Gere a memória real como uma **cache** para todas as imagens dos processos/ficheiros em disco
- Memória virtual (VM) – separação da memória lógica (virtual) da memória física
  - MV definida pelo espaço de endereçamento lógico (ou virtuais)
  - Só parte da imagem do processo precisa de estar em memória
  - Um processo pode ter mapa de memória **>** memória física
  - É possível ter memória física **<** soma das imagens de todos os processos
  - Todas as páginas dos processos podem ser guardadas em memória secundária, tipicamente chamado disco/ficheiro de paginação ou de swap
  - As páginas em swap só voltam para memória central quando são novamente referenciadas