Sistemas Operacionais

Gestão de memória - Tópicos avançados

Prof. Carlos Maziero

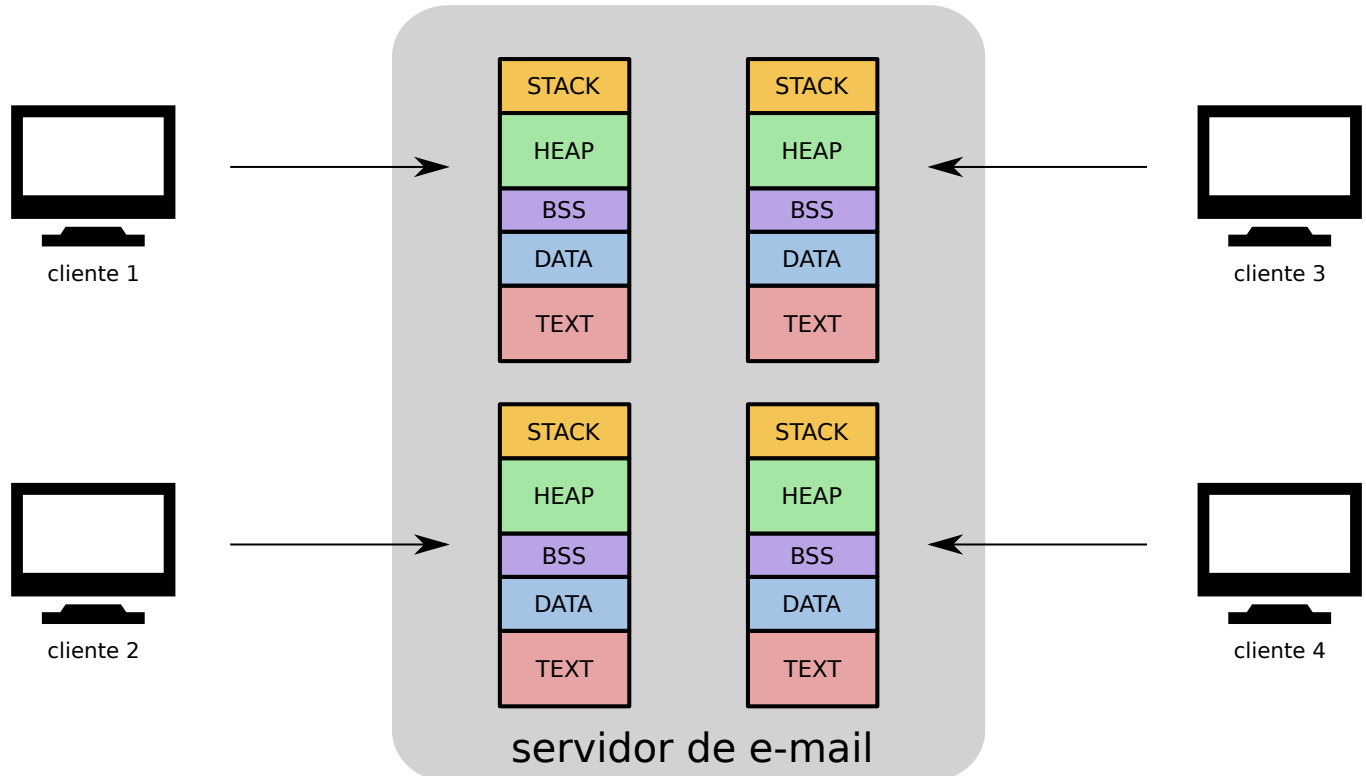
DInf UFPR, Curitiba PR

Agosto de 2020

Conteúdo

Desperdício de memória

- Processos distintos executando o mesmo código
- Mesmo conteúdo replicado: desperdício de RAM!



Compartilhamento de memória

Situação:

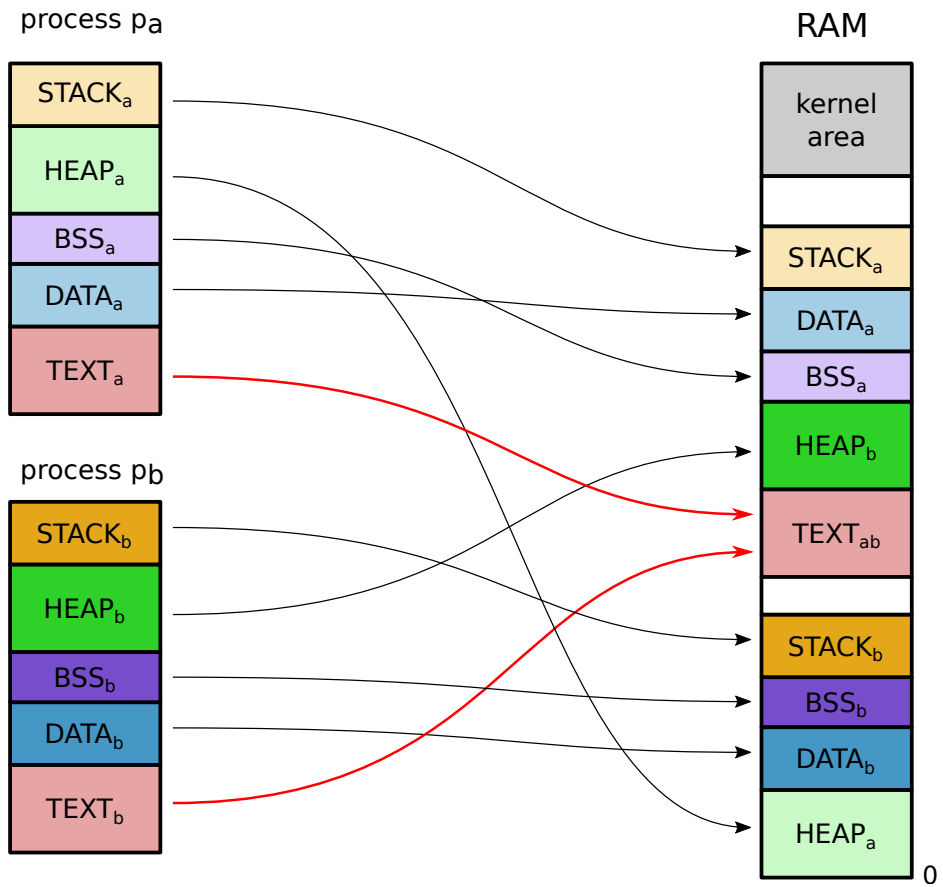
- Processos com o mesmo código têm o mesmo `TEXT`
- O mesmo ocorre com áreas de constantes e bibliotecas
- Essas áreas têm conteúdo fixo (*read-only*)

Proposta:

- Compartilhar seções de memória *read-only*
- Alocar seções idênticas na mesma área de RAM
- Mapear essa área na memória virtual dos processos

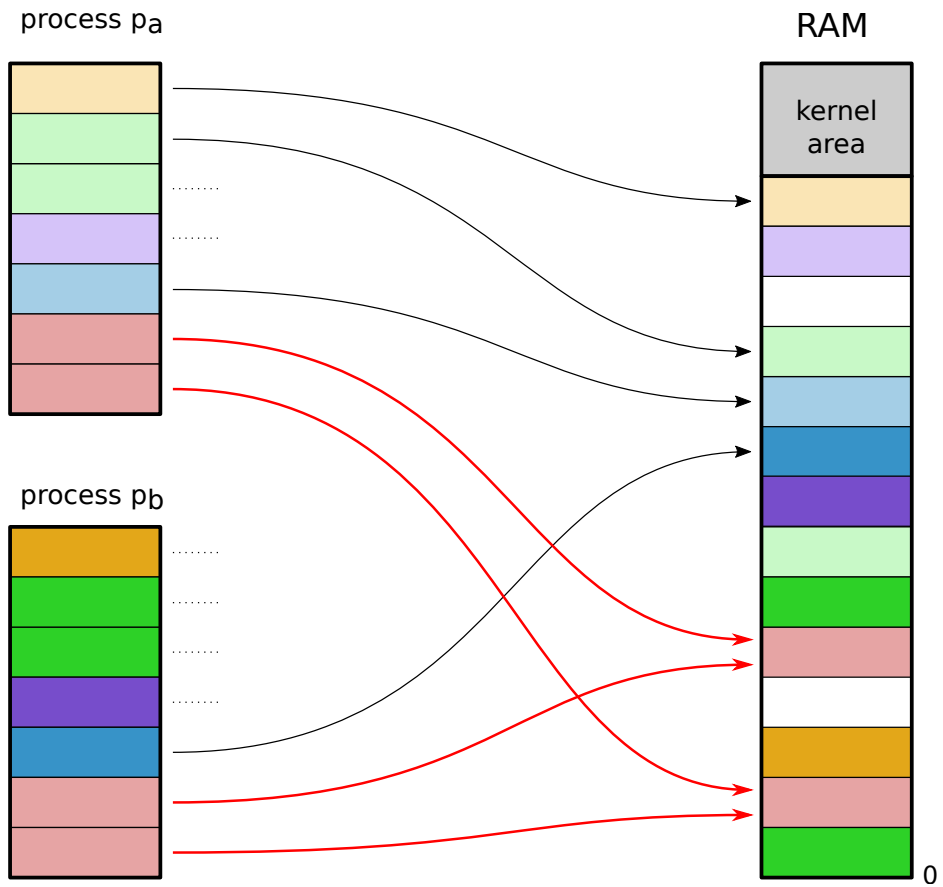
Compartilhamento de segmentos

Segmentos *read-only* apontam para mesma RAM:



Compartilhamento de páginas

Páginas *read-only* apontam para os mesmos quadros:



Benefícios do compartilhamento

a) **Economia** de memória RAM:

- Conteúdo fixo é carregado uma só vez em RAM
- Áreas de código, constantes e bibliotecas

Ex.: 10 processos iguais, 60MB de código + 40MB de dados:

- Sem compartilhamento: $10 \times (60 + 40) = 1000MB$
- Com compartilhamento: $60 + 10 \times 40 = 460MB$

b) Mecanismo eficiente de **comunicação** entre processos

Copy-on-write (CoW)

Forma de compartilhamento de RAM mais "agressiva":

- Compartilha **tudo** (inclusive páginas *read-write*)
- Quando houver uma escrita em página compartilhada, desfaz o compartilhamento
- Usa flags *Copy-on-Write* (cow) na tabela de páginas

Muito usada na chamada `fork()`:

- Filho compartilha **todas** as páginas do processo-pai
- Páginas do pai e do filho são marcadas *read-only* e *cow*
- Compartilhamento é desfeito sob demanda

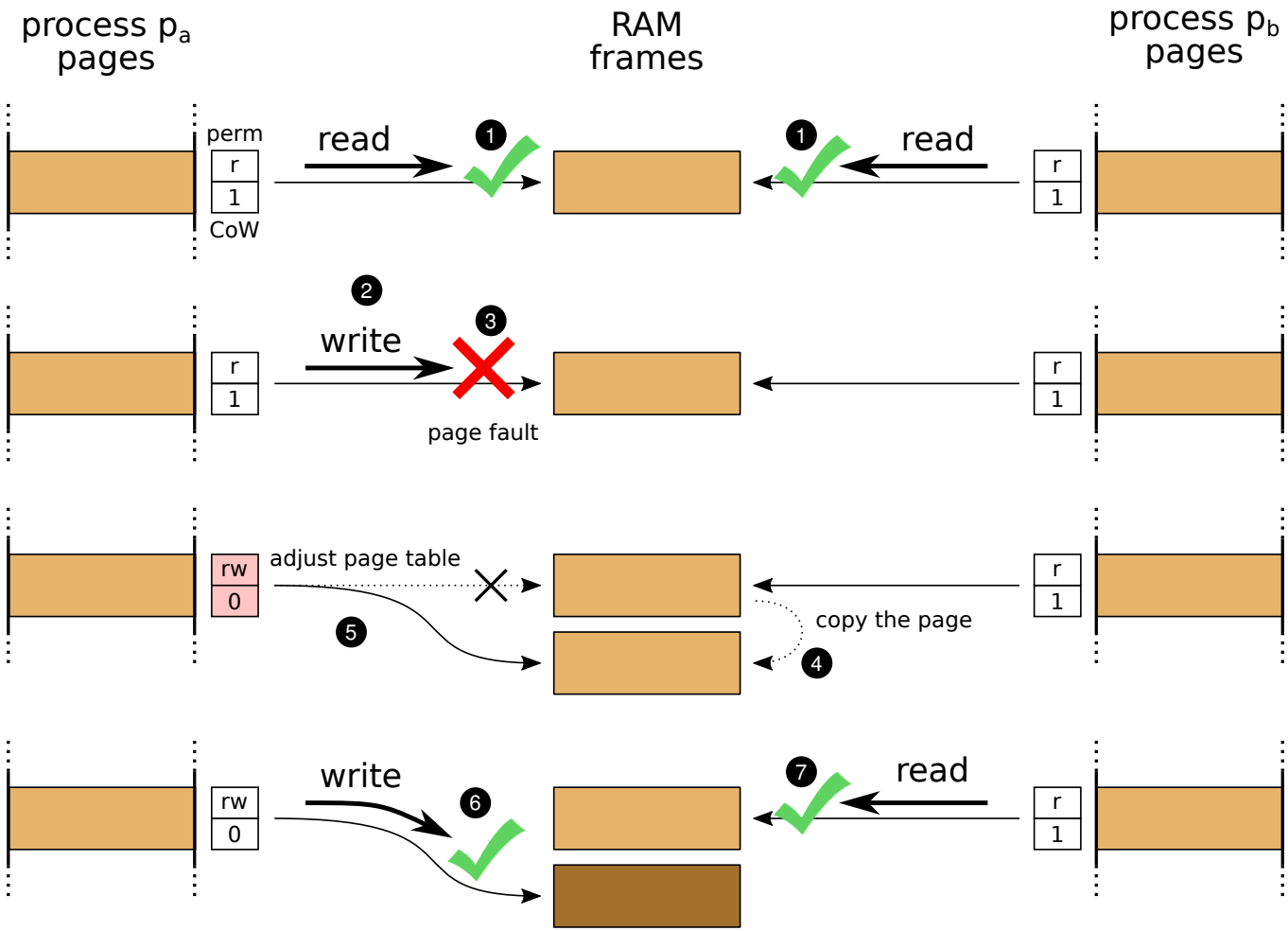
Copy-on-write (CoW)

Exemplo: p_a e p_b são dois processos (pai e filho)

Funcionamento:

1. p_a e p_b compartilham páginas em CoW; leitura está ok
2. p_a tenta escrever em uma página compartilhada
3. a MMU nega o acesso e gera uma *page fault*
4. o núcleo cópia a página em outro quadro da RAM
5. o núcleo ajusta a tabela de páginas de p_a e os flags
6. p_a retoma a execução e completa a escrita
7. p_b continua a acessar a página original

Copy-on-Write (CoW)



Arquivo mapeado em memória

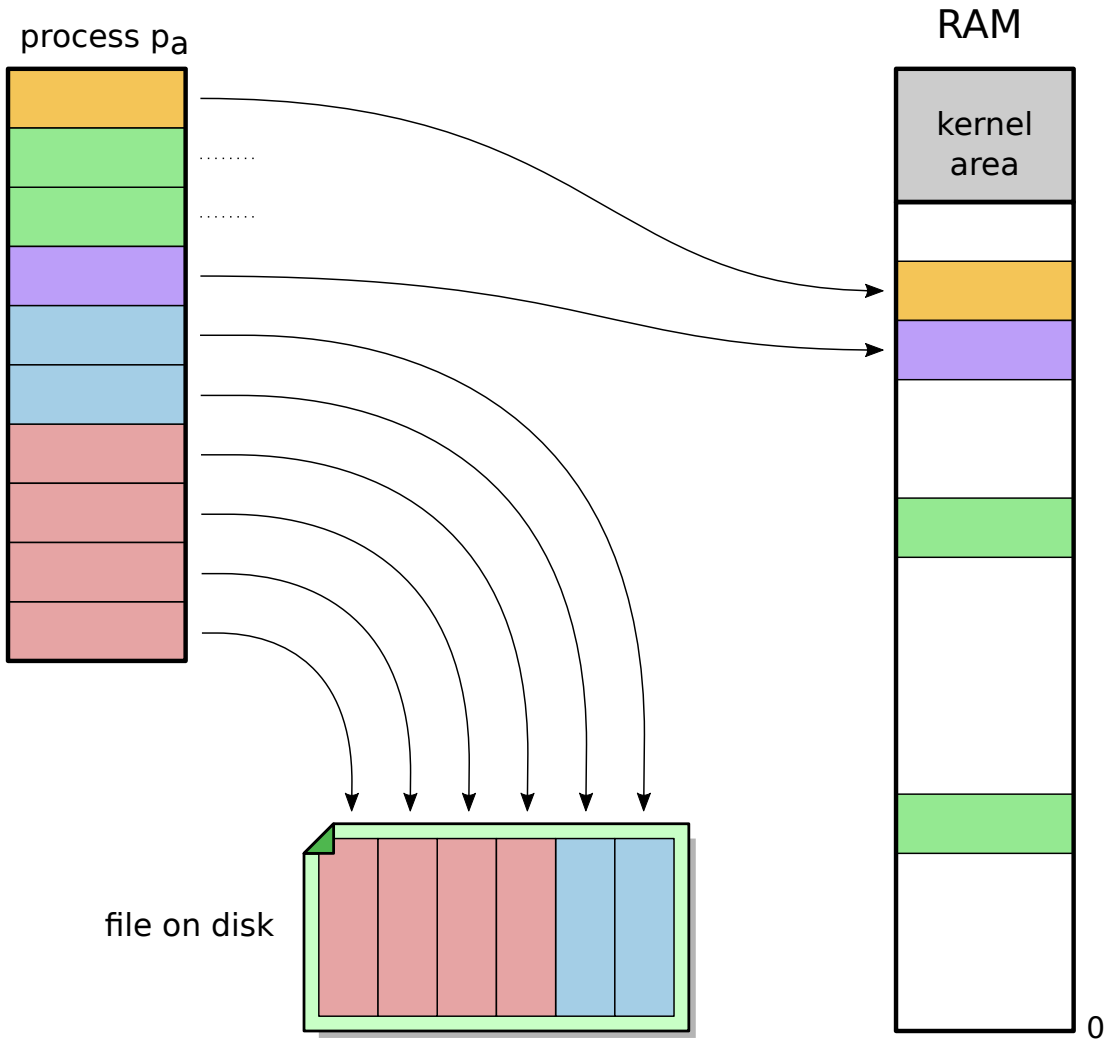
Associa uma área do processo a um arquivo em disco:

- Cada byte da área corresponde a um byte do arquivo
- Ao ler da área na memória, o núcleo busca do arquivo
- Ao escrever na área, o núcleo escreve no arquivo
- O mapeamento pode ser *read-only* ou *read-write*

Vantagens:

- Agiliza o acesso a arquivos
- Não precisa carregar o arquivo inteiro na memória
- Muito usado com arquivos executáveis e bibliotecas

Arquivo mapeado em memória



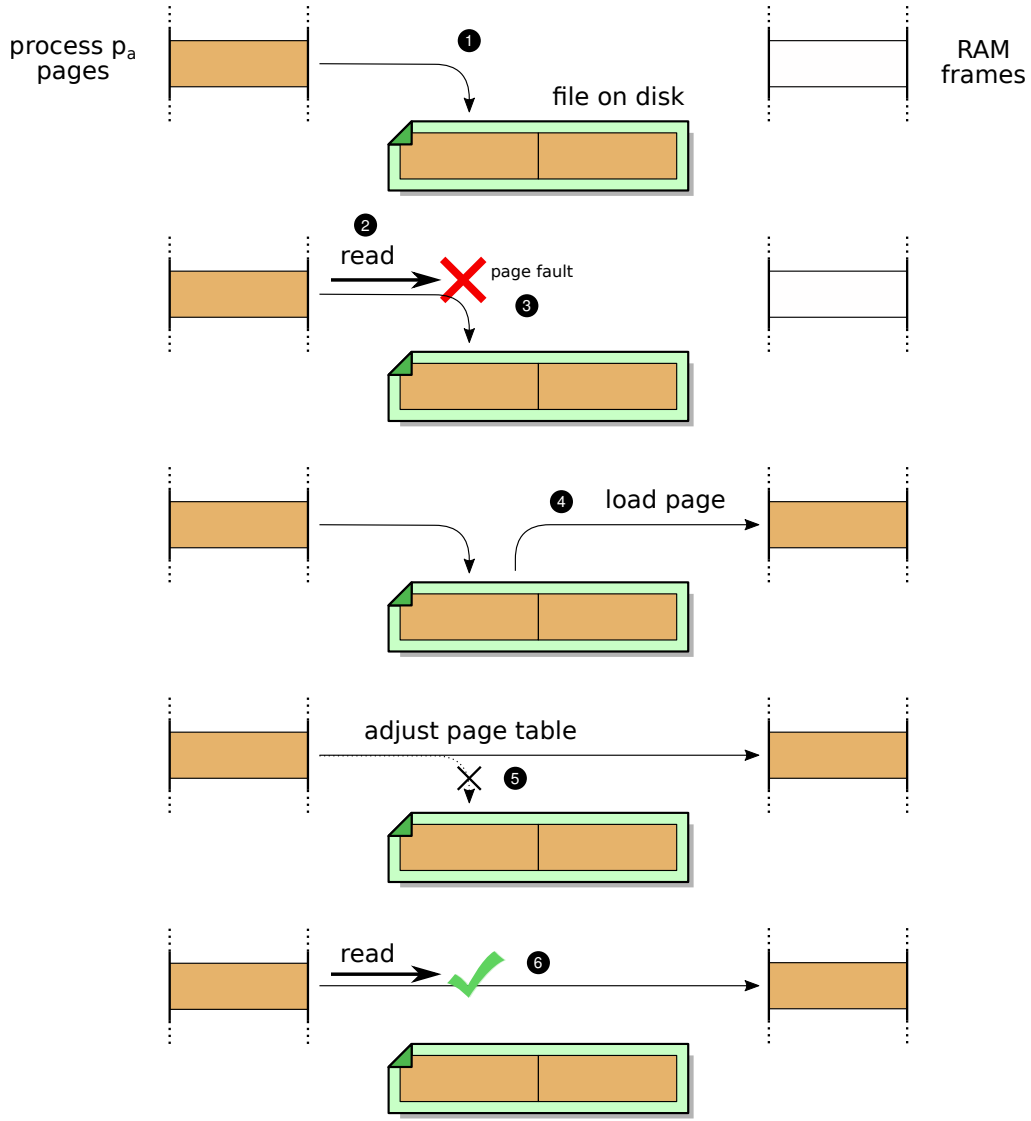
Paginação sob demanda

Mecanismo usado com o mapeamento em memória

Páginas do arquivo mapeado são carregadas ao acessá-las:

1. Um arquivo é mapeado em uma área do processo
2. O processo tenta ler um byte dessa área
3. Como a área ainda não foi carregada, ocorre um *page fault*
4. O núcleo carrega aquela parte do arquivo na memória
5. O núcleo ajusta a tabela de páginas do processo
6. O processo retoma a execução e acessa a posição da RAM

Paginação sob demanda



Tópicos avançados a abordar

- Lazy memory allocation
- Garbage collection
- Dynamic linking libraries
- Carga de processo e bibliotecas
- ...