MAC0422 - Sistemas Operacionais

Daniel Macêdo Batista

IME - USP, 26 de Outubro de 2020

Roteiro

Implementação de diretórios

Gerência de espaço livre

Implementação de diretórios

Gerência de espaço livre

Implementação de diretórios
Gerência de espaço livre

Implementação de diretórios

Implementação de diretórios

Implementação de diretórios

Gerência de espaço livre

- A principal função de um diretório é mapear o nome do arquivo para alguma informação necessária para localizar os dados desse arquivo (primeiro bloco + quantidade de blocos, primeiro bloco ou número do i-node)
- Logo, precisa armazenar as informações dos arquivos que estão "abaixo" dele

Onde armazenar os atributos dos arquivos?

- Diretamente no diretório
  1. Um diretório é uma lista com 1 entrada para cada arquivo do diretório
  2. Cada entrada na lista tem um tamanho fixo (nome do arquivo, estrutura dos atributos do arquivo e um ou mais endereços do disco informando onde os dados estão)

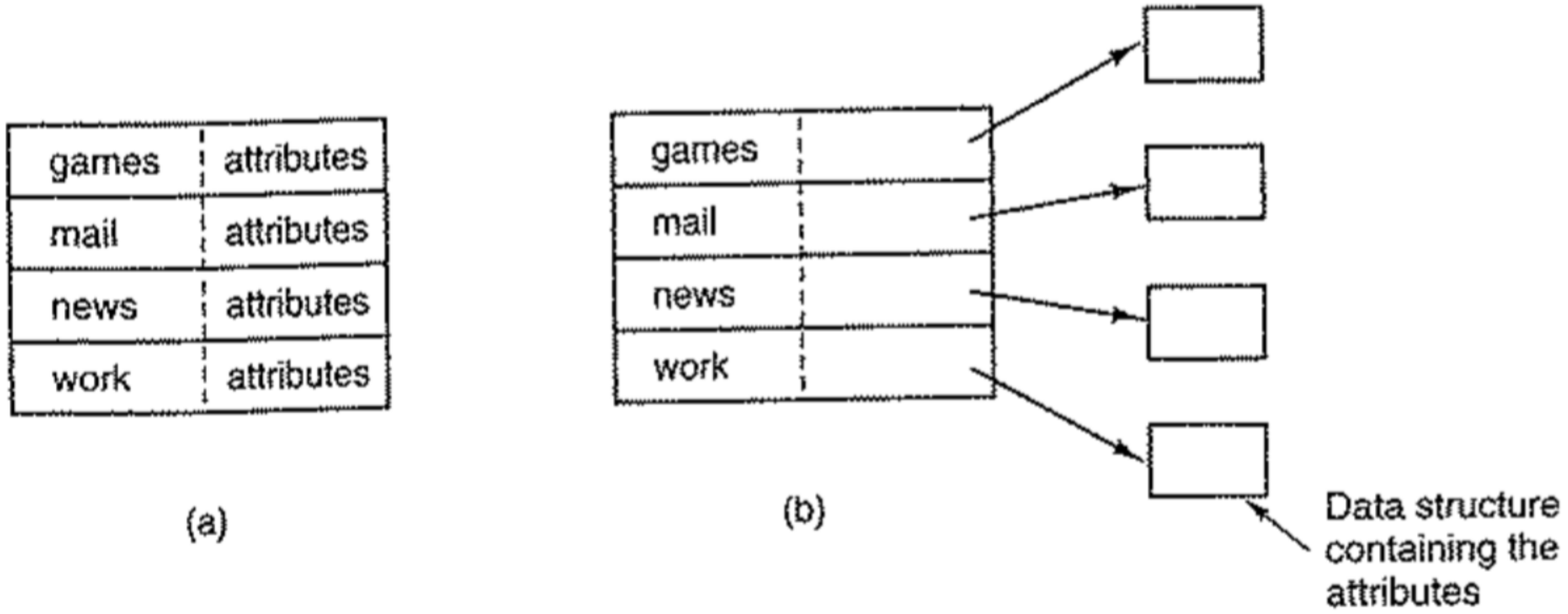
Onde armazenar os atributos dos arquivos?

- Nos i-nodes (se for armazenamento com i-nodes claro)
  1. Um diretório é uma lista com 1 entrada menor do que no caso anterior pois tem apenas o nome do arquivo e o número do i-node

Onde armazenar os atributos dos arquivos?

Implementação de diretórios

Gerência de espaço livre



Como guardar os nomes?

- Lembrando que é necessário haver um limite no tamanho dos nomes dos arquivos (antigamente, 8.3 caracteres, atualmente comum ser 255 caracteres)
- Pode usar alguma das estruturas do slide anterior fixando o nome do arquivo em 255 caracteres mesmo que não ocupe tudo isso (ruim porque raramente um arquivo tem nome tão grande)
- O melhor é considerar tamanho variável e armazenar os dados considerando que eles estão contíguos no diretório (in-line) ou considerando que há ponteiros para os dados de cada arquivo (heap)

Guardando nomes in-line

- Cada entrada contém o tamanho da entrada seguido dos dados em um formato pré-estabelecido (O nome do arquivo é o primeiro dado depois do tamanho)
- Pode ser necessário manter os nomes múltiplos de alguma quantidade de bytes (múltiplo de 4 bytes por exemplo)
- A desvantagem é que se um arquivo é removido pode ser que o espaço liberado por ele não seja suficiente para um próximo arquivo (mas compactar agora é mais fácil porque quando um diretório é aberto ele fica todo em memória)
- Outra desvantagem é que uma entrada do diretório pode estar em várias páginas da memória e uma falha de páginas pode ocorrer quando um nome de arquivo estiver sendo lido

Guardando nomes na heap

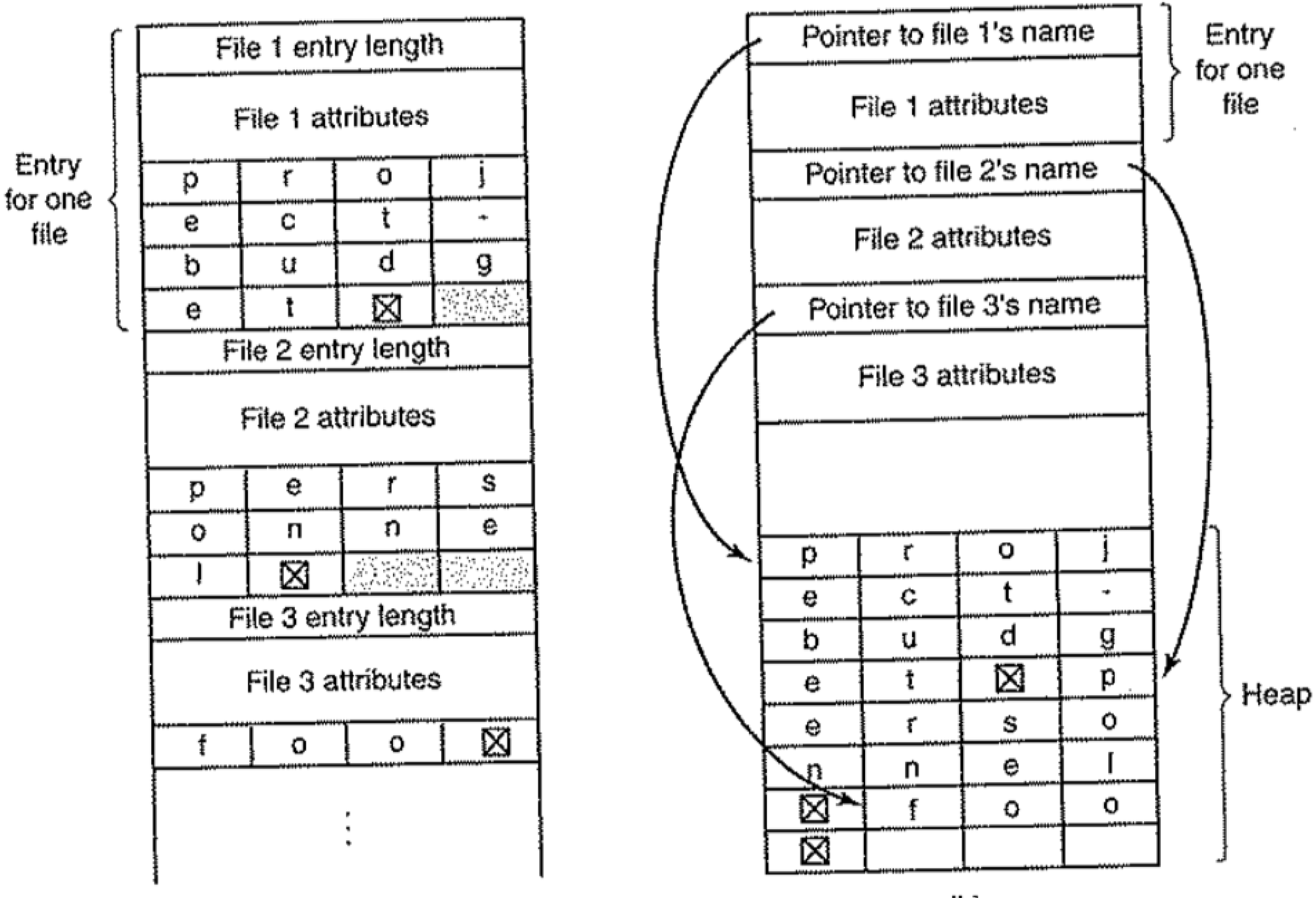
- Cada entrada contém um ponteiro para o nome do arquivo seguido dos atributos do arquivo
- Os nomes dos arquivos ficam no fim do diretório em uma heap

- Cada entrada agora consegue então ter um tamanho fixo
- A vantagem em relação ao caso anterior é que se um arquivo é removido o espaço liberado será suficiente para outro arquivo (claro que a heap precisa ser gerenciada)
- Outra vantagem é que os tamanhos dos nomes dos arquivos não precisam ser múltiplos de nenhum valor

Resumo do armazenamento de nomes nos diretórios

Implementação de diretórios

Gerência de espaço livre



Gerência de espaço livre

Como saber quais blocos estão livres?

- Precisa armazenar de alguma forma os blocos livres
- Opções:
  - Bitmap
  - Lista encadeada

Bitmap

Implementação de diretórios

Gerência de espaço livre

- Um disco com n blocos precisa de um bitmap com n bits.
- Blocos livres podem ser representados por 1 e os blocos alocados podem ser representados por 0
- Um disco de 500GB com blocos de 1KB tem cerca de 488 milhões de blocos $\rightarrow$ O bitmap terá 488 milhões de bits (vai usar cerca de 60 mil blocos de 1KB para poder ser armazenado)

Lista encadeada

- A ideia é ter uma lista encadeada apontando para os blocos livres
- Geralmente os próprios blocos livres são usados para guardar os números dos blocos que estão livres (mas precisaria no início da partição ter pelo menos 1 ponteiro para o primeiro bloco livre da lista)

  Ex.: um bloco de 1KB possui espaço para armazenar 1024*8 bits = 8192 bits. Se o disco possui 500GB, e usa 32 bits para endereçar todos os blocos, nesse caso, em cada bloco é possível guardar 8192/32 endereços de blocos = 256 blocos, mas 1 desses grupos de 32 bits tem que apontar para o próximo bloco livre, então o total é 255 blocos em cada bloco

Lista encadeada melhorada

- Pode ser melhorado para o caso de haver vários blocos consecutivos. Nesse caso cada bloco livre teria associado um outro valor com a contagem de blocos livres a partir dele
- Mas isso vai ser ruim se o disco estiver muito fragmentado (decisão difícil)

Lista encadeada X bitmap

- Bitmaps ocupam menos espaço na maioria das vezes mas o espaço ocupado é sempre fixo
- Lista encadeada fica mais eficiente quando o disco enche pois o espaço ocupado é proporcional ao espaço livre (mas como pode ser mantido nos blocos livres, é como se não ocupasse tanto espaço assim)

Quotas

Implementação de diretórios

Gerência de espaço livre

- SOs usados por múltiplos usuários precisam de mecanismos para impedir que um único usuário monopolize todo o sistema de arquivos. Sistemas de quotas existem para isso
- Importante lembrar que não bastar limitar o espaço em disco mas também a quantidade de arquivos (i-nodes por exemplo)
- Obviamente para isso funcionar é necessário que o proprietário de um arquivo seja armazenado nos atributos do mesmo e mudanças nos tamanhos dos arquivos sejam atualizados em uma tabela de usuários

Quotas

- Na maioria das vezes há dois tipos de limites: *soft* e *hard*
- Passar do limite *soft* faz o sistema gerar alertas para o usuário mas ele ainda pode criar mais arquivos ou usar mais espaço. Passar do limite *hard* impedirá o usuário de criar novos arquivos ou usar mais espaço