Exercício Programa 3

Simulador de sistema de arquivos

Lucas Paiolla Forastiere, 11221911

Marcos Siolin Martins, 11221709

IME-USP

07 de dezembro de 2020

Detalhes de Implementação - o arquivo

- A representação do sistema de arquivos é armazenada em um arquivo que sempre ocupa `100MB` no sistema de arquivos real. Caso seja executado `mount` sobre um arquivo que não exista, será gerado um novo arquivo com `100MB` onde estará armazenado o `Bitmap`, a FAT e o diretório `root` (/);
- Consideramos que conteúdo de arquivos contém apenas caracteres que ocupam 1 `byte` em seus nomes e conteúdos, ou seja, que pertencem à tabela `ASCII`. Isso nos permite controlar quanto espaço cada arquivo ou diretório ocupa;

Detalhes de Implementação - o arquivo

- Utilizamos o caractere | (pipe) como separador para indicar situações como o fim de nome de arquivo ou fim de conteúdo, então é importante que não existam arquivos que contenham esse caractere no nome ou em seu conteúdo;
- Para preencher espaços em branco utilizamos a constante `CHAR_NULO` que é um ' ' (whitespace).
- Após dar mount no arquivo que guarda o sistema de arquivos simulado, o conteúdo do arquivo é trazido para memória e as alterações são feitas em memória. As alterações serão gravados no arquivo em disco quando o comando umount for dado.

Detalhes de Implementação - o bitmap

- O Bitmap é implementado como um vetor booleano de tamanho `NUM_BLOCOS`, que é a constante que guarda a quantidade de blocos disponíveis para o sistema de arquivos simulado, desconsiderando os blocos necessários para armazenar o Bitmap e a FAT. O valor `1/true` indica que o bloco está livre e o valor `0/false` indica que o bloco está ocupado.
- O Bitmap ocupa os primeiros 7 blocos do sistema de arquivos simulado, pois precisa armazenar `NUM_BLOCOS bytes`. O espaço restante no 7º bloco é desperdiçado;

Detalhes de Implementação - a FAT

- A FAT é implementada como um vetor de inteiros de tamanho `NUM_BLOCOS`. O valor em `ponteiro[i]` indica qual é o próximo bloco após o *i* na lista ligada do arquivo. Caso esse valor seja igual à `BLOCO_NULO` (um valor de um bloco que não existe), então o bloco *i* é o último na sequência da lista ligada.
- Para armazenar esses ponteiros os convertemos para uma string com tamanho fixo 5, assim, se `ponteiro[i] = 1`, no sistema de arquivos simulado será armazenado como `00001`.
- A FAT é armazenada nos 32 blocos conseguintes ao Bitmap, pois precisa armazenar `NUM_BLOCOS*5 bytes`. O espaço restante no 32º bloco é desperdiçado;

Detalhes de Implementação - o root

- O diretório / é um diretório especial. Ele está sempre ocupando o bloco 0 (a partir de agora desconsideraremos os blocos necessário para armazenar o bitmap e a FAT) e também armazena os próprios metadados, nessa ordem:
  - Tempo Criado - ocupa 10 bytes. É a quantidade em segundos devolvida por `time(NULL)` no momento de criação do arquivo;
  - Tempo Modificado - ocupa 10 bytes. É a quantidade em segundos devolvida por `time(NULL)` no momento de última modificação do arquivo;
  - Tempo Acesso - ocupa 10 bytes. É a quantidade em segundos devolvida por `time(NULL)` no momento de último acesso do arquivo;
  - Nome - ocupa x bytes. Ao fim do nome estará o caractere '|'.

Detalhes de Implementação - os diretórios

- Os diretórios armazenam os metadados dos subdiretórios e dos arquivos que estão "imediatamente abaixo" do diretório. Os metadados são armazenados na seguinte ordem:
  - Ponteiro para o nome - ocupa 8 bytes. Aponta para o endereço do disco onde está o nome do arquivo/diretório;
  - Diretório - ocupa 1 byte. Indica se os metadados são de um diretório ou de um arquivo;
  - Número do primeiro bloco - ocupa 5 bytes. Aponta para o primeiro bloco onde o arquivo/diretório está armazenado;
  - Tempo Criado - ocupa 10 bytes;
  - Tempo Modificado - ocupa 10 bytes;
  - Tempo Acesso - ocupa 10 bytes;

Detalhes de Implementação - os diretórios

  - Tamanho - ocupa 8 bytes. Se os metadados são de um diretório, então esse valor é sempre 0;
  - Nome* - ocupa x bytes. Ao fim do nome estará o caractere '|'.

- Os diretórios seguem a estratégia apresentada em aula onde cada subarquivo/subdiretório tem um campo de metadados, com exceção do nome (*), de tamanho fixo e o metadado nome fica armazenado em uma região especial chamada de heap. No campo de metadados existe um ponteiro para o lugar onde o nome está armazenado;
- Na nossa implementação a heap está armazenada imediatamente após acabarem todos os campos para os metadados. O começo da heap é indicado pelo caractere '|'.