EP3 - MAC0422

Diego Alvarez, Thiago I. S. Pereira

22 de novembro de 2015

Introdução

Esta apresentação tem como objetivo exemplificar a implementação do EP3 de MAC0422 - Sistemas Operacionais.
Iremos apresentar um simulador de sistema de arquivos baseado em FAT, usando bitmaps para gerencia do espaco livre.

O Simulador

O simulador foi totalmente implementado e testado ultilizando python3 em ambiente GNU/Linux

- python3 - versão 3.4.2
- Debian GNU/Linux 8.2 Jessie

O Simulador

O simulador e o shell foram implementados separadamente, o shell chamando o simulador como se fosse um programa separado, cada um tendo seus proprios módulos.

O Simulador

O simulador foi dividos nos seguintes modulos:

- **interface** - Implementa os comandos pedidos no enunciado, se utilizando das outras estruturas para controlar. É o módulo de mais alto nível.
- **filesystem** - Implementa a gerencia das estruturas principais (FAT e bitmap) e intermedia o acesso ao arquivo físico. É o modulo de mais baixo nível.
- **directory** - Estrutura que corresponde a um diretorio no sistema de arquivos e possui facilidade para manipular suas entradas
- **entry** - Estrutura de uma entrada de em diretorio. Mantem os metadados de arquivos e diretorios, sempre estão contidos em um diretorio.

O Simulador

O shell foi dividido nos seguintes modulos:

- **prompt** - Verifica a entrada do usuário, se todos os comandos e valores correspondem com o enunciado do EP.
- **main** - Inicia um shell e com a ajuda do modulo prompt verifica as entradas e executa o simulador com os parametros inseridos

Formato binário

- O sistema de arquivos é composto por 24.986 setores, a FAT ocupa 49.972 bytes e o bitmap 3.124 bytes
- O arquivo binário guarda primeiro o bitmap, seguido pela FAT, seguido pelos setores para armazenamento
- O diretório raiz sempre está nos 3 primeiros setores

Formato binário

- Os diretorios são formados por uma lista que ocupa 3 blocos (12KB) e comportam um numero fixo de 240 entradas
- Cada entrada tem um tamanho de 50 bytes e é composta de 31B para o nome, 1B para o tipo, 4B para o tamanho, 3x4B para as datas e 2B de ponteiro para os dados
- Assim cada arqivo/diretório deve ter no máximo 31 bytes (em utf-8) de nome

Metodologia

- Foi feito um script para rodar todos os testes automaticamente
- Foram rodados os 8 testes pedidos para cada um dos estados (vazio, 10MB cheio e 50MB cheio)
- Cada teste foi repetido 30 vezes e seus resultados estão a seguir

Testes

Tempo em milisegundos

Teste	Vazio	10MB cheio	50MB cheio
cp 1mb	7.6	13.3	36.3
cp 10mb	44.2	98.8	335.7
cp 30mb	292.6	408.1	1160.3
rm 1mb	6.8	6.5	6.4
rm 10mb	6.8	6.8	6.9
rm 30mb	7.8	7.7	8.1
rmdir vazio	35.7	34.1	34.4
rmdir cheio	40.4	40.2	41.1

Análise

- Mesmo as operacoes simples demoram um pouco devido ao overhead de desmontar o sistema de arquivos
- A operacao mais lenta é de copiar arquivos, o que é esperado pois há mais dados a serem movidos
- O estado do sistema de arquivos aparentemente só influencia a velocidade de cópia, o que faz sentido pois essa é a unica operacao que precisa alocar espaco
- Remover uma arvore de diretorios cheia é apenas um pouco mais lento que remover uma vazia