Trabalho de Sistemas Operacionais
Trabalho P3

Entrega 21/06/2017
Prof. Carlos R. Moratelli

Roteiro

1. Objetivos

1. Implementação de um sistema de arquivos simplificado, baseado em FAT.
2. Reforço no aprendizado sobre implementação de sistema de arquivos.
3. Aprendizado de manipulação de arquivos e estruturas em dispositivo de armazenamento secundário.

2. Descrição do sistema de arquivos

Este trabalho consiste na implementação de um simulador de sistema de arquivos. Um programa desenvolvido pelo aluno, em C, deve manipular estruturas de dados em um arquivo simulando o comportamento de um sistema de arquivos em disco. A implementação deve seguir a especificação a seguir:

- O sistema de arquivos utiliza método de alocação de blocos encadeado (conforme explicado em aula).
- O sistema de arquivos deve considerar setores de 512 bytes.
- O diretório raiz (root) fica no setor 0 e além de uma lista de arquivos e diretórios ele armazena um ponteiro para a lista de blocos livres.
- Os setores são numerados de 0 a n.
    - O sistema de arquivos utiliza 4 bytes (32 bits) para numeração dos blocos. Assim, são possíveis 2^{32} blocos de 512 bytes cada, totalizando 2 terabytes de espaço total suportado pelo sistema de arquivos.
    - O tamanho do arquivo é armazenado em uma variável de 32bits. Portanto, um arquivo pode ter tamanho máximo de 4GB.
- O sistema de arquivos utiliza mapeamento de blocos livres por encadeamento.
    - O diretório raiz guarda o ponteiro para o primeiro bloco livre.

As estruturas abaixo definem o formato dos dados a serem utilizados:

```
/**
 * File or directory entry.
 */
struct file_dir_entry{
	unsigned int dir;
	char name[20];
	unsigned int size_bytes;
	unsigned int sector_start;
};

/**
 * Root directory table.
 * First directory table of the file system. Should be written to the sector
0.
 */
struct root_table_directory{
	unsigned int free_sectors_list;
	struct file_dir_entry entries[15];
	unsigned char not_used[28];
};

/**
 * Subdirectories file table.
 */
struct table_directory{
	struct file_dir_entry entries[16];
};

/**
 * Sector data.
 */
struct sector_data{
	unsigned char data[508];
	unsigned int next_sector;
};
```

2.1 Detalhes sobre o sistema de arquivos

- Se uma entrada de diretório (`file_dir_entry`) possui *name* igual a string vazia (“”) significa fim da lista de arquivos.
- Quando a estrutura `file_dir_entry` apresentar apenas start como 0, significa que aquele arquivo ou diretório foi excluído e seu nome não deve ser apresentando para o usuário, a entrada fica disponível para um novo arquivo.
- Se o membro next_sector de sector_data for 0, significa fim da lista de blocos para o

arquivo em questão.

- A lista de blocos livres é criada atualizando a variável free_blocks_list, no diretório raiz, com o valor do primeiro bloco livre. Assim, cada bloco deve apontar para o próximo bloco livre. O número do próximo bloco livre (4 bytes) deve estar gravado conforme a struct sector_data. Se o valor do membro next_sector for 0, significa que não existem mais blocos livres.

3. Sobre a implementação.

A implementação deve usar como base os fontes disponíveis no GitHub, conforme explicado em sala. Para obter os fontes use o comando git, exemplo:

```
git clone https://github.com/crmoratelli/fs_simul.git
```

A aplicação deve suportar as seguintes operações sobre o sistema de arquivos aqui descrito:

- Inicializar
  - exemplo: simulfs -format
- Criar (-create <arquivo original> <destino no sistema virtual>)
  - exemplo: simulfs -create /home/user/classe.xls /<caminho>/alunos.xls
- Ler (-read <arquivo de saída> <caminho no sistema virtual>)
  - exemplo: simulfs -read /home/user/classe.xls /<caminho>/alunos.xls
- Apagar
  - exemplo: simulfs -del <caminho>/aluno.xls
- Listar arquivos ou diretórios.
  - exemplo: simulfs -ls <caminho>
    - f paisagem.jpg 2048 bytes
    - d viagem
    - f lobo.jpg 5128 bytes
- Criar diretório
  - exemplo: simulfs -mkdir <caminho>/aulas
- Apagar diretório - Somente apaga se o diretório estiver vazio.
  - exemplo: simulfs -rmdir <caminho>/aulas

Nota: <*caminho*> refere-se ao diretório onde a ação será realizada. Cada nome de diretório é separado por “/”. Exemplo:

/home/user/aulas

O diretório raiz é indicado apenas por “/”. Por exemplo, listar os arquivos e diretórios do

diretório raiz:

simulfs -ls /

3. Entrega

O trabalho deve ser realizado em grupos de no máximo 3 alunos. Junto com a implementação, deve ser entregue (no Moodle) um relatório (no máximo 3 páginas) explicando detalhes da implementação. A entrega do trabalho deve ser realizada pelo Moodle. Durante a apresentação, todos os participantes devem estar presentes. A conferência do trabalho se dará através de um script e arquivos de teste previamente fornecidos pelo professor. Poderá ser usado a ferramenta md5sum para verificar a integridade dos arquivos.

3.1 Critérios de Pontuação

A pontuação do trabalho será dada conforme a tabela abaixo:

Item/Comando	Pontuação
Inicializar	1
Criar	1
Ler	1
Apagar	1
Listar arquivos ou diretórios	1
Criar diretório	2
Apagar diretório	1
Consistência/qualidade da implementação.	1
Apresentação	1

Nota: Ambos os comandos *ler/criar* devem estar funcionando para a pontuação valer. Não vale dizer que o comando *criar* foi implementado se o comando *ler* não estiver funcionando. São exigidos os dois comandos para fins de teste.