Trabalho de Sistemas Operacionais
Trabalho Prático 2

Prof. Carlos R. Moratelli
Entrega 15/06/2016

Roteiro

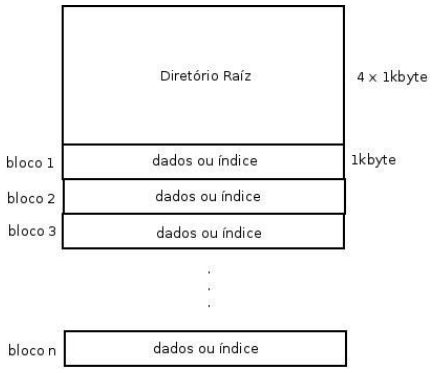
1. Objetivos

1. Implementação de um sistema de arquivos simplificado.
2. Reforço no aprendizado de sistema de arquivos.

2. Desenvolvimento

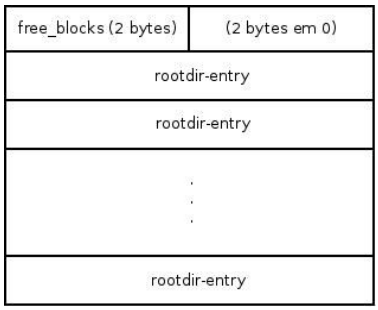
Este trabalho consiste na implementação de um simulador de sistema de arquivos. Um programa desenvolvido pelo aluno, em C, deve manipular estruturas de dados em um arquivo simulando o comportamento de um sistema de arquivos em disco. A implementação deve seguir a risca a especificação a seguir.

O sistema de arquivos armazena arquivos de maneira indexada (conforme explicado em aula). Os blocos livres são marcados através de uma lista encadeada de blocos livres. Por simplicidade, o sistema de arquivos não contará com subdiretórios, consistindo apenas de um diretório raiz. O diretório raiz deve armazenar uma lista de nomes de arquivos, tamanho e o ponteiro para o bloco de índice do arquivo. Serão reservados 4 blocos para o diretório raíz. Cada bloco deverá ter 1024 bytes. Abaixo a estrutura básica do sistema de arquivos.



Os blocos são numerados de 1 a n. O sistema de arquivos utiliza 2 bytes (16 bits) para numeração dos blocos. Assim, são possíveis 65536 blocos de 1024 bytes cada, totalizando 64Mbytes suportados.

Os primeiros 2 bytes (16 bits) do diretório raíz consistem em um ponteiro para o primeiro bloco livre. Os próximos dois bytes são sempre 0. O restante consiste nas entradas de diretório formadas por nome (20 bytes incluindo terminador null), tamanho 2 bytes (16 bits) e ponteiro para o bloco de índice 2 bytes (16 bits). Abaixo o diretório raiz em detalhes.



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

- Estrutura para cada entrada da tabela de diretórios.

```
struct rootdir_entry{
        char name[20];
        unsigned short int size;
        unsigned short int index;
};
```

- Estrutura do rootdir.

```
struct rootdir{
        unsigned short int free_blocks
        unsigned short int trailing;
        struct rootdir_entry list_entry[170];
        unsigned char reserved[12];
};
```

- Bloco de dados de dados:

```
unsigned char sector_data[1024];
```

- Bloco de índice:

```
unsigned short int sector_index[512];
```

Quando uma entrada de diretório (rootdir_entry) possui nome, size e index iguais a 0, significa

fim da lista de arquivos. Se apenas index for 0, significa que aquele arquivo foi excluído e seu nome nao deve ser apresentando para o usuário, a entrada fica disponível para um novo arquivo. Quando uma entrada de setor de índice for 0 significa fim da lista de blocos para o arquivo em questão.

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

- Inicializar
    - exemplo: simulfs -init
- Criar
    - exemplo: simulfs -create alunos.xls
- Ler
    - exemplo: simulfs -read alunos.xls
- Apagar;
    - exemplo: simulfs -del aluno.xls
- Listar arquivos.
    - exemplo: simulfs -ls

O arquivo gerado pela aplicação deve chamar-se simul.fs. **Inicializar** o sistema de arquivos cria a estrutura do diretório raíz, se o arquivo simul.fs já existir ele deve ser zerado. Nenhum bloco vazio deve ser alocado, assim, o ponteiro de blocos livres deve ser inicializado em 0. O arquivo simul.fs deve assumir o seu tamanho máximo em sua inicialização. Sempre que um arquivo for apagado a lista de blocos livres deve ser atualizada. **Criar** um arquivo consiste em ler um arquivo no sistema de arquivos real e gravá-lo no sistema de arquivos simulado. Sempre devem ser utilizados primeiro os blocos livres, caso não existam blocos livres suficientes, deve-se alocar novos blocos até o total mencionado anteriormente. **Ler** consiste em ler o arquivo do sistema de arquivos simulado gravando-o no sistema de arquivos real. Implemente as 5 operações seguindo a sintaxe dos exemplos acima.

A lista de blocos livres é cria atualizando a variável free_blocks com o valor do primeiro bloco livre. Assim, cada bloco deve apontar para o próximo bloco livre. O número do próximo bloco livre (2 bytes) deve estar gravado no início do bloco. Se o seu valor for 0, significa que não existem mais blocos livres.

Para implementação devem ser utilizadas as primitivas de manipulação de arquivos binários da linguagem C: fopen(), fwrite(), fread(), fseek() e fclose(). **Não será permitido carregar todo o sistema de arquivo para memória para manipulação das estruturas de dados.** Sempre que algum dado na tabela de diretórios seja alterado, ela deve ser salva. Utilize a ferramenta md5sum para ter certeza que os arquivos criados e lidos a partir do seu sistema de arquivos continuam consistentes.

3. Entrega

O trabalho deve ser realizado com no máximo 2 alunos, para exceções consultar o professor. Junto com a implementação, deve ser entregue um relatório de no máximo 3 páginas explicando detalhes da implementação. A entrega do trabalho deve ser realizada pelo Moodle e o mesmo deve ser apresentado em sala de aula na data indicada no cronograma da disciplina. Durante a apresentação, todos os participantes devem estar presentes. A conferência do trabalho se dará através de uma sequencia de operações (inicializar, criar, ler, apagar e listar) com arquivos fornecidos pelo professor.