Universidade de Brasília – UnB
Instituto de Ciências Exatas – IE
Departamento de Ciência da Computação – CIC

Trabalho

Disciplina: *Organização de Arquivos – Turma A*

Semestre: 2º/2017

Prof.: Patrick Pascoal de Brito Silva

Título: *Simulação de Disco Magnético Rígido*

Entrega: 28/11/2017 – 23h55 (Arquivos-Fontes via e-mail para patrickpascoalbs@gmail.com, num único arquivo zipado. Incluir um comentário no início com os nomes e matriculas dos alunos. O arquivo zipado deve ser a matrícula do líder do grupo, ex.: 1612345.zip)

1. Objetivos

O objetivo deste trabalho é que o aluno aprenda a estrutura do disco magnético e como são organizados os arquivos dentro dele.

Implementar, utilizando obrigatoriamente a linguagem C ou C++.

2. Especificação

2.1 Ambiente:

Crie uma estrutura de dados para emular um disco magnético rígido (HD) com as seguintes características:

- Trilhas por cilindro: 5
- Setores por trilha: 60
- Trilhas por superfície: 10
- Tamanho do setor: 512 bytes
- Tamanho do cluster: 4 setores
- Tempo médio de seek: 4 ms
- Tempo mínimo de seek: 1ms
- Tempo media de latência: 6ms
- Tempo de transferência: 12ms/trilha

Para isso no programa principal deve ser utilizado um ponteiro para indicar os cilindros e alocar a quantidade de cilindros necessária.

```
track_array *cylinder;
```

Devem ser definidas como tipos globais, as seguintes estruturas:

```
typedef struct block { unsigned char bytes_s[512]; } block;
typedef struct sector_array { block sector[60]; } sector_array;
typedef struct track_array { sector_array track[5]; } track_array;
```

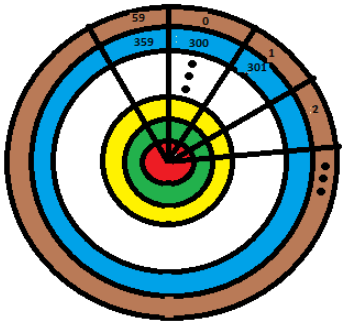
Deve ser criada também uma tabela FAT. A tabela FAT deve ser feita a partir de dois arrays de registros. O primeiro array deve conter uma lista com os nomes dos arquivos e o indicador do primeiro setor onde o arquivo esta armazenado no HD.

```
typedef struct fatlist_s {
char file_name[100];
unsigned int first_sector;
} fatlist;
```

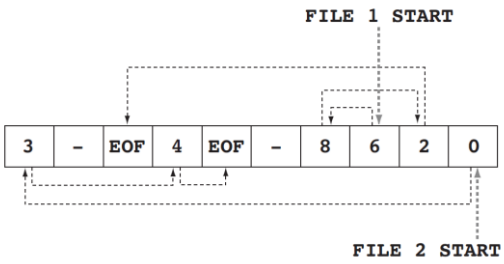
O segundo array deve ser uma lista com uma entrada por cada setor do HD.

```
typedef struct fatent_s {
unsigned int used;
unsigned int eof;
unsigned int next;
} fatent;
```

Sendo que o elemento 0 do array representa o primeiro setor da primeira trilha do primeiro cilindro. O elemento 1 é o segundo setor da primeira trilha do primeiro cilindro. O elemento 60 é o primeiro setor da segunda trilha do primeiro cilindro. O elemento 300 é primeiro setor da primeira trilha do segundo cilindro. Conforme a figura abaixo.



O campo 'used' indica se o setor esta livre ou sendo utilizado. O campo 'eof' indica se é o último setor do arquivo. O campo 'next' indica o próximo setor do arquivo. Um exemplo de como a FAT deve funcionar está mostrada na Figura abaixo:



2.2 Software:

O programa deve ter como nome: 'my_drive.cpp'. E assim que for executado deve mostrar o seguinte menu:

```
1 - Escrever Arquivo
2 - Ler Arquivo
3 - Apagar Arquivo
4 - Mostrar Tabela FAT
5 - Sair
```

Opção (1)

O programa deve pedir ao usuário o nome do arquivo (o qual deve ser um arquivo de TEXTO (.TXT) na mesma pasta onde está sendo executado o programa). O arquivo deve ser lido à memória principal, dividido em um número inteiro de clusters, e salvo do HD fictício. Da seguinte forma:

- Achar o primeiro cluster disponível na primeira trilha do primeiro cilindro. Se não houver nenhum cluster disponível, passar para a primeira trilha do próximo cilindro. Caso, todas as primeiras trilhas de todos os cilindros estiverem cheias, então passar para a segunda trilha do primeiro cilindro, e assim por diante.

- Uma vez que achou um cluster disponível, gravar o primeiro cluster do arquivo nesse local, inserir o mesmo na FAT na lista de nomes. O próximo cluster no arquivo deve ser gravado na mesma posição na próxima trilha do cilindro (a partir do mesmo número de setor da trilha seguinte). Caso essa posição esteja sendo usada passar para a próxima trilha, e assim por diante. Caso todas as outras posições do cilindro estejam sendo usadas, passar para o próximo cluster. Desta forma, estamos emulando a escrita em paralelo de um cilindro.

- A cada cluster gravado deve-se atualizar a lista de setores utilizados da FAT. Indicando que esta em uso (colocando '1' no indicador 'used').

- No final deve indicar a quantidade de tempo utilizado para gravar o arquivo (assumir seek médio para achar o primeiro cilindro).

Opção (2)

O programa deve pedir um nome de arquivo (TXT) e procurar pelo mesmo na tabela FAT. Se não encontrar, indicar arquivo inválido. Senão deve ler o arquivo, o que significa juntar os cluster e escrever o arquivo no **HD REAL** com o nome: SAIDA.TXT. Indicar o tempo de leitura (assumir seek médio para achar o primeiro cilindro).

Opção (3)

Deve indicar em cada setor do arquivo na lista de setores da FAT, que esse setor esta livre (colocando '0' no indicador 'used'). E retirar o arquivo da lista de nomes da FAT.

Opção (4)

Mostrar a tabela FAT no seguinte formato:

```
NOME:                TAMANHO EM DISCO    LOCALIZAÇÃO
ARQUIVO1.TXT         2048 Bytes          0,1,2,3
ARQUIVO2.TXT         4092 Bytes          4,5,6,7,64,65,66,67
ARQUIVO3.TXT         6144 Bytes          8,9,10,11,68,69,70,71,128,129,130,131
.
.
.
ARQUIVON.TXT         2048 Bytes          300,301,302,303
```

Onde a localização é o número dos **setores** ocupados pelo arquivo.

3. Avaliação

Cada uma das quatro opções do menu terão **20% da nota total do trabalho e a avaliação do código terá 20% da nota total**. Os alunos deverão mostrar o programa funcionando no dia marcado para a entrega do trabalho (28/11/2017).

4. Grupos

Neste projeto será permitido a formação de grupos de no máximo **três** alunos. A partir do grupo formado, deverá ser indicado **um líder que será o responsável pelo envio dos arquivos fontes** por e-mail para o professor. Somente serão aceitos os arquivos fontes enviado pelo líder do grupo.

5. Observações Importantes

- Quanto melhor comentado o código, mais bem avaliado ele será.
- Trabalho identificado como **cópia** entre alunos ou de **algoritmos da Internet** (qualquer uma das partes) recebe a nota ZERO.
- Os trabalhos devem seguir boas práticas de programação: Identação, não declarar variáveis no meio do código, comentários, etc.
- Não serão aceitos trabalhos atrasados **EM HIPÓTESE ALGUMA**.