

## Laboratório 6: Implementação do Sistemas de Arquivos

Em um certo SO, o disco rígido está organizado conforma Fig. 1:

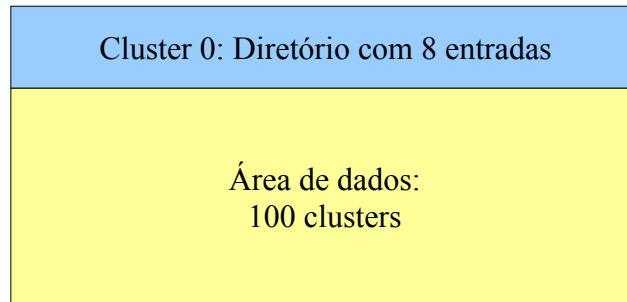


Fig. 1: Organização do disco

Cada entrada no diretório é feito utilizando as seguintes estruturas:

```
typedef struct
```

```
{
    unsigned short dia;
    unsigned short mes;
    unsigned short ano;
    unsigned short hora;
    unsigned short min;
    unsigned short seg;
}data;
```

```
typedef struct
```

```
{
    char nome[8]; //nome do arquivo iniciado com o hexadecimal 0EBh são arquivos deletados
    char extensao[3];
    // Proteção
    unsigned short sistema; //arquivo do sistema operacional
    unsigned short hidden; //arquivo oculto
    unsigned short archived; //arquivo arquivado
    // Data e hora da criação
    data criacao;
    // Data do último acesso
    data acesso;
    // Tamanho
    unsigned long int tamanho;
    // Cluster inicial
    unsigned long int cluster;
}arquivo;
```

Os clusters de dados possuem a seguinte estrutura:

```
typedef struct
```

```
{
    char dados[512];
    unsigned long int prox;
} cluster;
```

Para acessar um arquivo no disco são necessários duas informações contidas na estrutura arquivo, o ***tamanho***

que é o tamanho do arquivo em bytes e **cluster** que é o número do cluster inicial, ou seja aponta para o primeiro cluster que deve ser lido. Cada cluster lido possui na sua estrutura um ponteiro para o próximo cluster, até encontrar o valor zero. A Fig. 2 apresenta um esquema da localização de um arquivo.

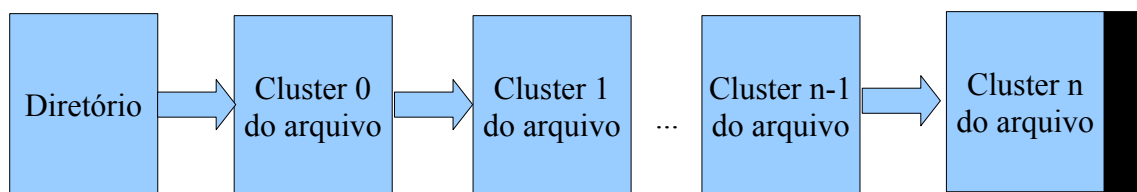


Fig. 2: Localização do arquivo no disco

Observando a Fig. 2 vemos uma parte preta no final do último cluster, é a parte não utilizada do cluster, esta é uma parte que não pode ser utilizada por nenhum outro arquivo. Podemos considerar fragmentação interna no cluster.

**Exercício 1:** Pede-se desenvolver um programa denominado **arq** que liste os arquivos armazenados neste arquivo, mas deverá seguir os seguintes critérios:

1. Se na execução do programa não for informado nenhum parâmetro o dir deverá imprimir somente o nome e a extensão no formato “nome.extensão”, com o ponto separando o nome da extensão e deverá imprimir 5 arquivos por linha igualmente espaçados e o nome do arquivo que possui o primeiro caracter o código ASCII 0xEB não deve ser impresso, pois é um arquivo **deletado** e nem os arquivos com proteção **sistema** ou **hidden** diferente de 0;
2. Se for utilizado o parâmetro **-a** (**arq -a**), a listagem imprimirá todos os dados dos arquivos, imprimindo 1 arquivo por linha, respeitando os mesmos critérios de seleção de arquivos do item anterior;
3. Se for utilizado o parâmetro **-s** (**arq -s**), o mesmo que o item 1, imprimindo inclusive os arquivos com os arquivos com proteção de sistema diferente de 0;
4. Se for utilizado o parâmetro **-w** (**arq -w**), listar somente os nomes e extensão dos arquivos, imprimindo 4 arquivos por linha distribuídos igualmente na linha, não deverão ser impressos arquivos escondidos, de sistema e deletados e
5. Qualquer outro parâmetro não conhecido imprimir a mensagem “Opção inválida para arq” e não listar mais nada.

**Exercício 2:** Pede-se desenvolver um programa em linguagem C denominado **type** que localize e imprima um arquivo, no caso o arquivo será o *Ext4.txt* (linha de comando: **type Ext4.txt**), dentro de Disco.dat, que é um arquivo binário simulando um disco rígido, este arquivo segue junto com o exercício e possui 52.520 bytes. Imprima somente o conteúdo do arquivo, no último cluster encontramos informações que não são do arquivo. Após a impressão de Ext4.txt imprima a sequência dos clusters lidos, a quantidade de clusters, o tamanho do arquivo e a quantidade de bytes desperdiçado no último cluster.

Sugestão: quando trabalhamos com arquivos binários podemos utilizar recursos como a função `fseek()`.