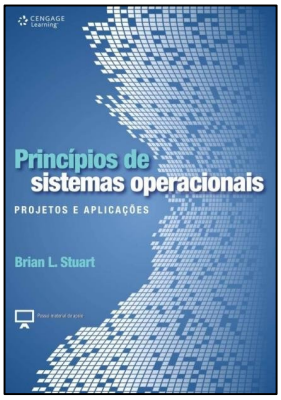
FACULDADE DE COMPUTAÇÃO E INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
SISTEMAS OPERACIONAIS – Aula 09 – 1º SEMESTRE/2020
PROF. LUCIANO SILVA

TEORIA: GERENCIADOR DE DISPOSITIVOS (PARTE I)

Nossos **objetivos** nesta aula são:

- conhecer a estrutura geral de um gerenciador de dispositivos
- conhecer como o Linux mapeia dispositivos de memória como arquivos
- detalhar o dispositivo de memória secundária disco, que dará suporte ao estudo do gerenciador de arquivos



Para esta aula, usamos como referência o **Capítulo 15 (Gerenciamento de Dispositivos)** do nosso livro-texto:

STUART, B.L., **Princípios de Sistemas Operacionais: Projetos e Aplicações**. São Paulo: Cengage Learning, 2011.

Não deixem de ler este capítulo depois desta aula!

GERENCIADOR DE DISPOSITIVOS

- Cada um destes gerenciadores é responsável por um tipo de objeto: **gerenciador de memória** (memória), **gerenciador de processos** (programas em execução), **gerenciador de dispositivos** (teclados, vídeo, discos,...) e **gerenciador de arquivos** (dados e pastas armazenados em memória secundária como, por exemplo, arquivos em discos).

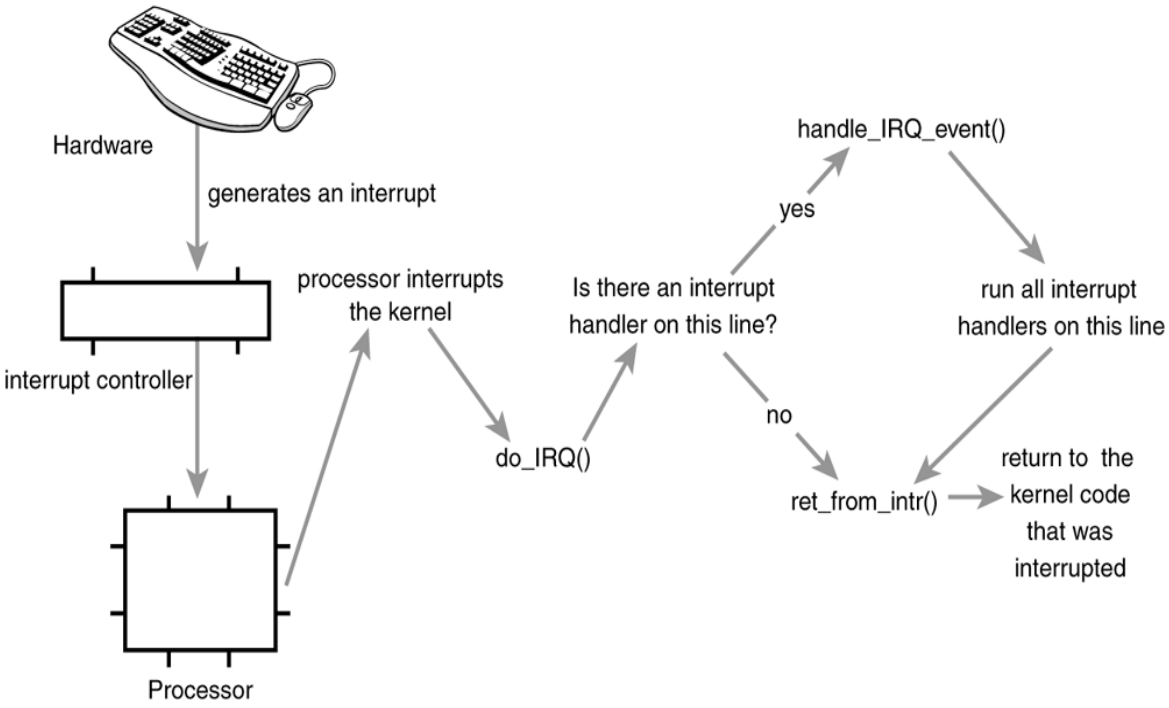
- O **gerenciador de dispositivos (ou gerenciador de E/S)**, é responsável pelo acesso a dispositivos de entrada/saída.

- O acesso a um dispositivo envolve:

    - operações realizadas no próprio hardware

    - auxílio de um **device driver**, já no nível de software e controlado pelo sistema operacional

- Embora existam várias técnicas de E/S, a mais comum é o mecanismo de **interrupção**, cujo esquema de funcionamento é mostrado abaixo:



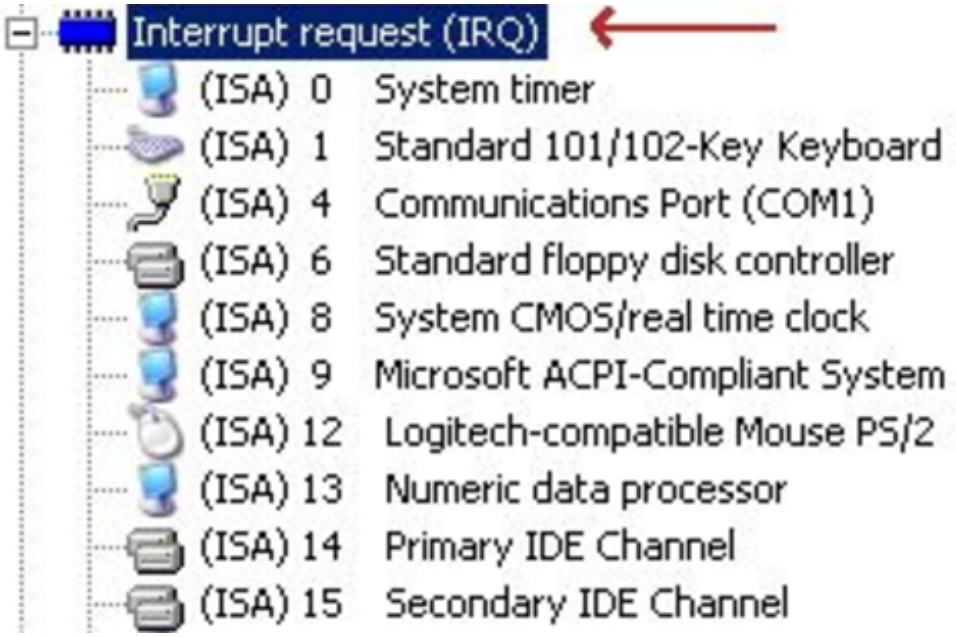
- No esquema acima, ocorre uma operação de entrada. Ao apertar uma tecla do teclado, geramos uma interrupção, que é capturada por um processador especializada chamado **controlador de interrupção** (*interrupt controller*). Este processador notifica a CPU de que há uma requisição de interrupção e, a CPU, envia esta solicitação para o gerenciador de dispositivos do SO.

- Estas requisições são chamadas de **IRQ** (Interrupt ReQuest). Cada dispositivo ou grupo de dispositivos possui um IRQ diferente para que o SO saiba quem está tentando se comunicar (ou com quem vai se comunicar).

- Há tabelas de IRQ padronizadas, como a mostrada abaixo:

IRQ	**Device**
0	System Timer
1	Keyboard Controller
2	Second IRQ Controller
3	COM 2
4	COM 1
5	Sound Circuit
6	Floppy Drive
7	Parallel Port
8	Real-time Clock
9	Available
10	Available
11	Available
12	Mouse Port
13	Math Coprocessor
14	Primary Hard Drive
15	Secondary Hard Drive

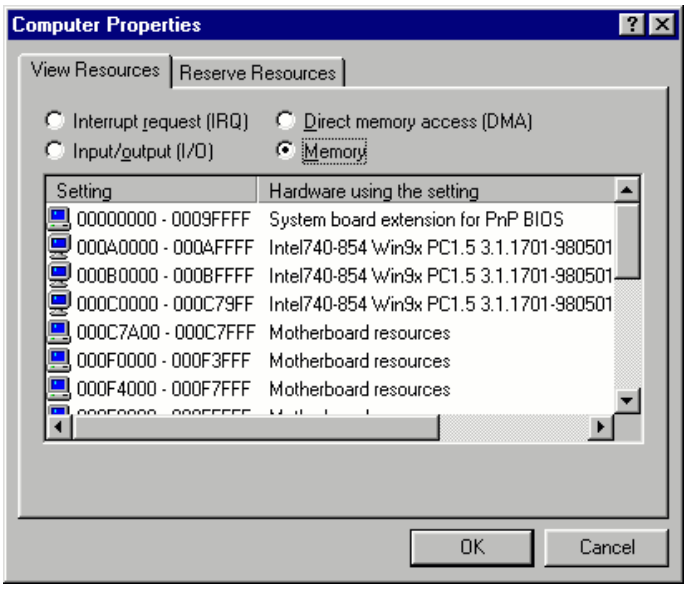
- Quando um SO está rodando, é possível também se consultar as IRQs. No exemplo abaixo, temos algumas IRQs para alguns dispositivos, mostrados no Sistema Operacional Windows:



- Observem que o mapeamento da IRQs no SO segue a tabela padronizada mostrada anteriormente.

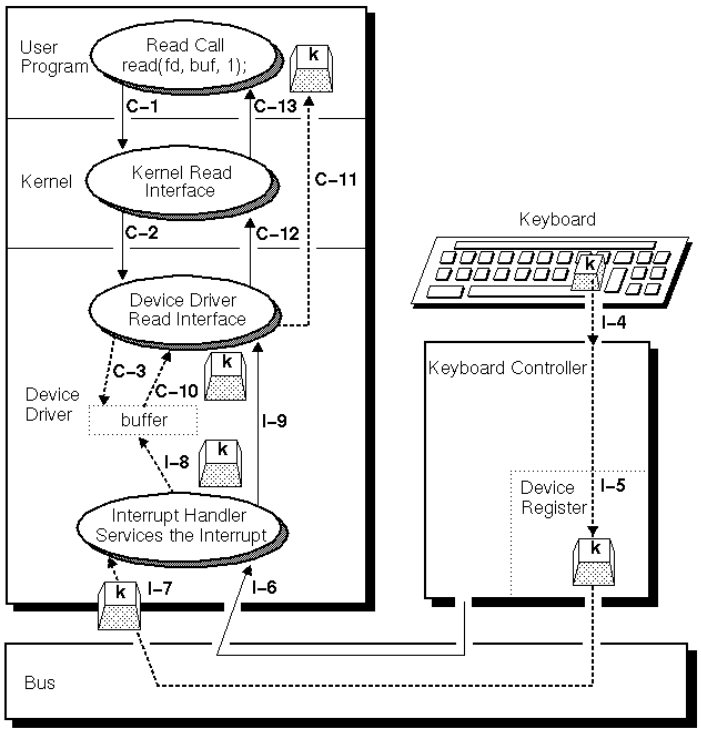
- Associada a uma IRQ, normalmente temos:

  - uma região de memória usada para comunicação (**buffer**) com o dispositivo. Esta área pode variar em função do tipo do dispositivo. A figura abaixo mostra exemplos destas regiões no Sistema Operacional Windows.



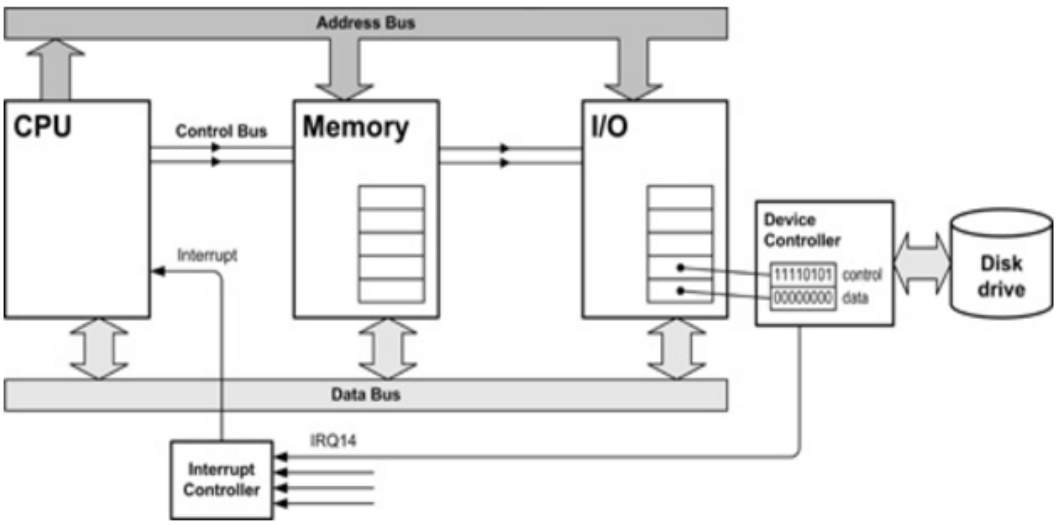
  - um programa de controle de acesso ao dispositivo, chamado **device driver**. Os device drivers, geralmente, são fornecidos pelo fabricante do dispositivo e contém as instruções de inicialização, controle, leitura e escrita no dispositivo. Device drivers são programados, geralmente, em linguagem C (médio nível) ou em Assembly.

- A figura abaixo ilustra o funcionamento do device driver com o restante do SO e hardware num comando do tipo **c=input()** .



- A comunicação da CPU não é feita, geralmente, diretamente com o dispositivo. Há uma elemento intermediário, chamado **controladora de dispositivo** (**device controller)**.

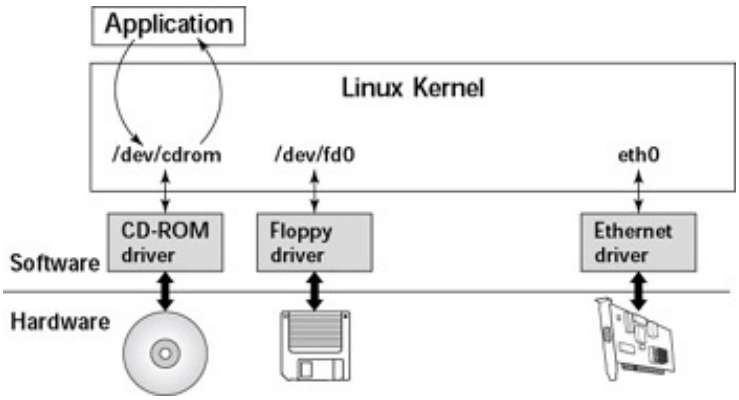
- As controladoras possuem memórias locais de comunicação com o dispositivo e, também, transformam as solicitações que vieram do SO, passaram pela CPU, em ações no dispositivo. Por exemplo, ler um arquivo armazenado em disco rígido.



- Abaixo, temos um exemplo de controladora ligada a um disco rígido.

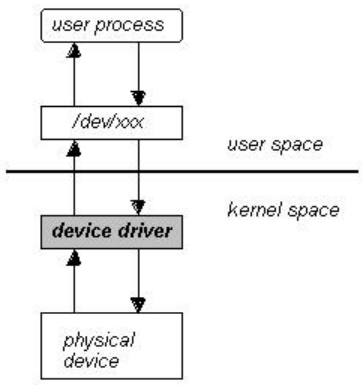
GERENCIAMENTO DE DISPOSITIVOS NO LINUX

- Dispositivos são “montados” no sistema de arquivos do Linux, a partir do diretório /dev.

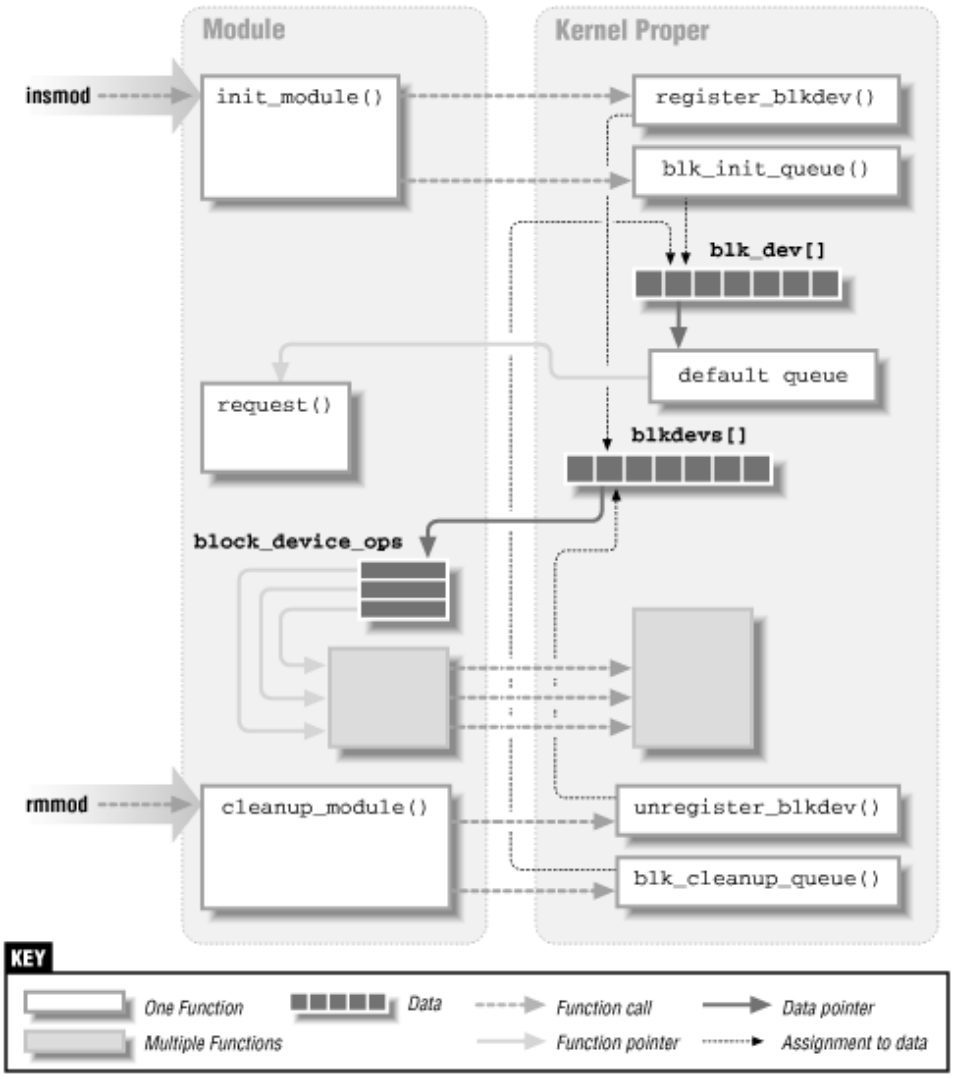


- Exemplos de pontos de montagem: /dev/hda1 (primeira partição do primeiro HD), /dev/hda2 (segunda partição do primeiro HD), /dev/cdrom (unidade de CDROM), /dev/fd0 (primeira unidade de disquete), /dev/eth0 (primeira placa de rede), /dev/eth1 (segunda placa de rede), /dev/tty (terminal).

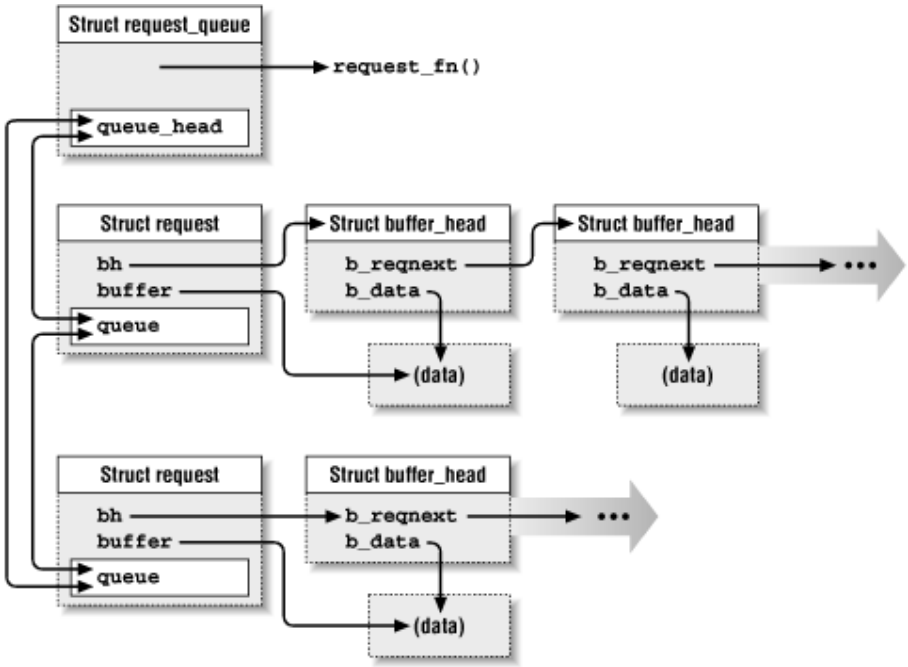
- Através do /dev/... temos acesso ao driver que controla o dispositivo.



- O registro e remoção de um device driver é mostrado na figura abaixo (dispositivo do tipo bloco – um HD, por exemplo):



- Durante os processos de leitura/escrita no dispositivo, o Linux utiliza uma série de i-nodes (sistema de arquivos em memória) para armazenar as informações:



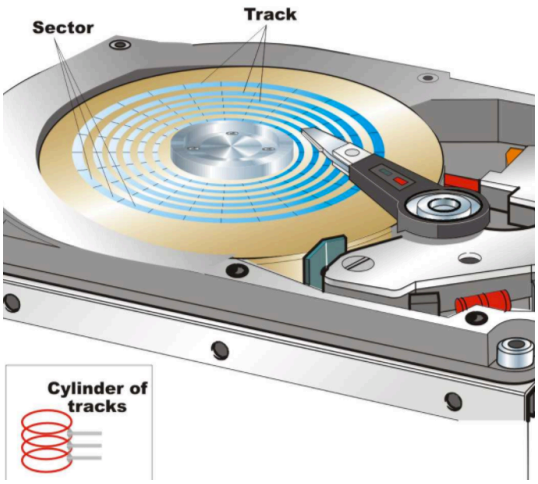
EXERCÍCIOS COM DISCUSSÃO EM DUPLAS

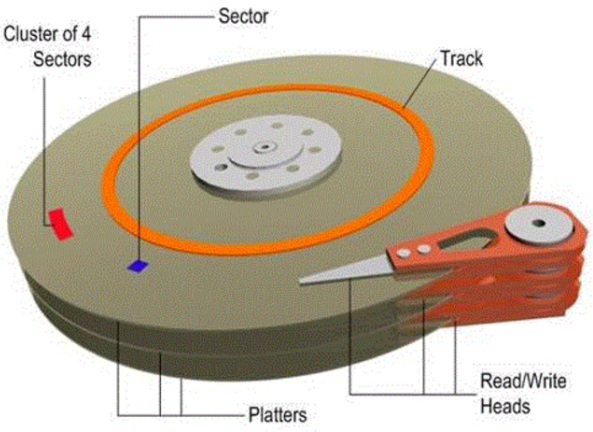
(a) Compare as diferenças de acesso a dispositivos de memória secundária pelos gerenciadores de dispositivos do Windows e do Linux.

(b) Chamamos de desmontagem o processo de descarregamento do buffers de I/O no dispositivo. Explique por que não se recomenda desconectar um pendrive montado em Linux sem o correto processo de desmontagem ?

ESTRUTURA DE DISCOS (MEMÓRIA SECUNDÁRIA)

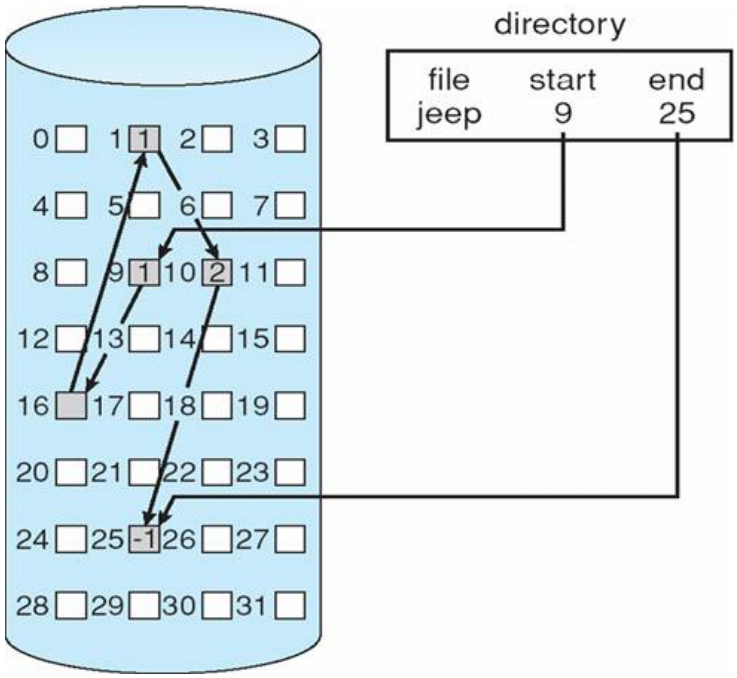
- Um disco rígido, por exemplo, pode ser descomposto em **setores** (sectors), **trilhas** (tracks) e **cilindros** (cylinders). Um setor é uma região do disco onde armazenamos os dados. Um tamanho típico de setor é 512 bytes. O conjunto de setores num caminho circular é chamado de trilha. Assim, trilhas são formadas de setores. Pequenos conjuntos de setores também podem formar **aglomerados** (clusters). Finalmente, trilhas de mesmo raio formam um cilindro, em discos que possuem múltiplos pratos de armazenamento.



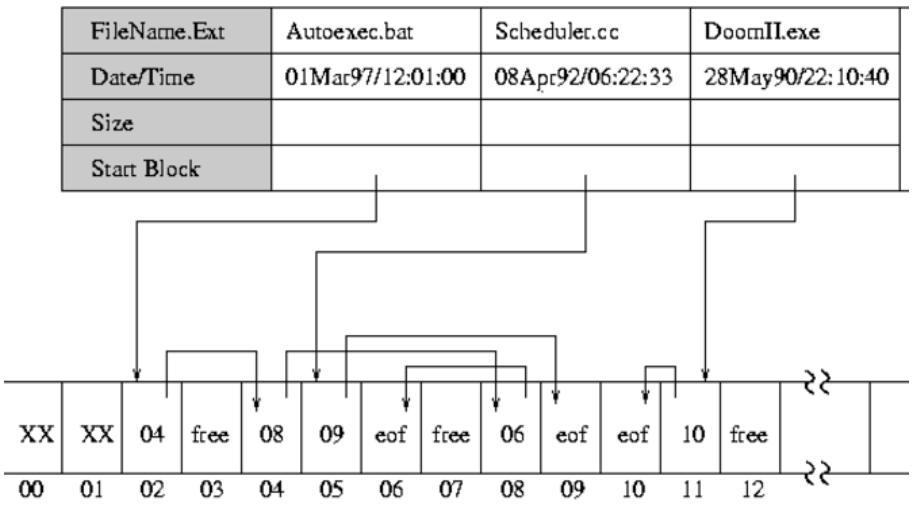


- Antes de realizarmos o armazenamento, o dispositivo de armazenamento precisa ser **formatado**. Na formatação, criamos um mapa de cilindros, trilhas e setores, que são é armazenado numa tabela do disco. Dependendo do tipo de sistema operacional, esta tabela pode mudar.

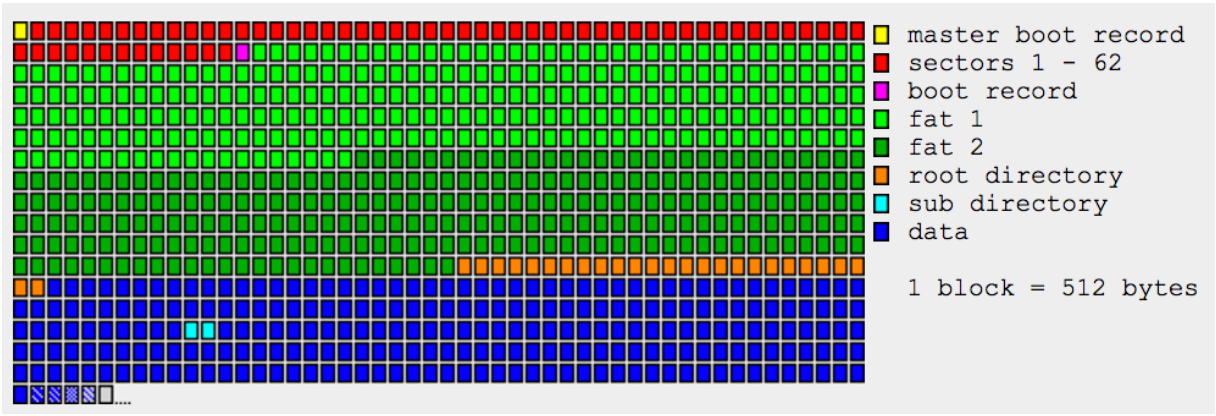
- Para o SO, um arquivo armazenado no disco é formado por **blocos** (blocks). Um bloco possui um endereço no disco (cilindro, trilha e setor(es)). Genericamente, um arquivo é armazenado como mostrado abaixo:



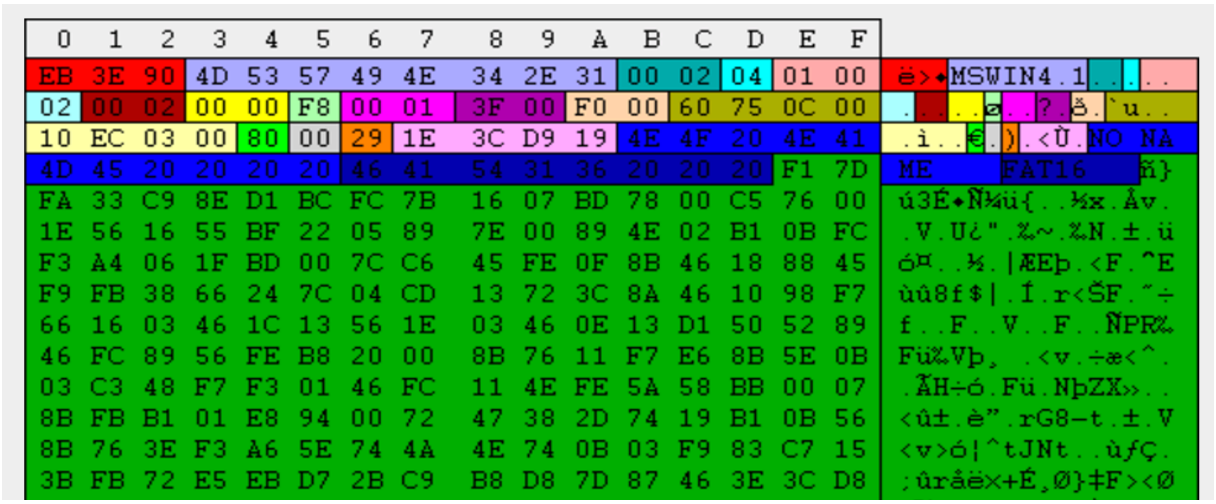
- Além de conter o nome do conjunto de blocos (nome do arquivo) e a localização dos blocos do arquivo, a tabela de arquivos também pode armazenar as propriedades do arquivo (data da criação/modificação, tamanho, permissões de acesso,...). Abaixo, temos parte da tabela FAT, usada pelos SOs DOS e Windows. Neste tipo de tabela, o final de arquivo é marcado com EOF (End Of File), ao invés de armazenar o bloco final.



- A FAT, por exemplo, é armazenada logo no início do disco. Outras tabelas de arquivos podem ser armazenadas em outros pontos do disco (meio, final), para otimizar o acesso. Abaixo, temos um exemplo de localização da FAT e arquivos alocados em um disco.

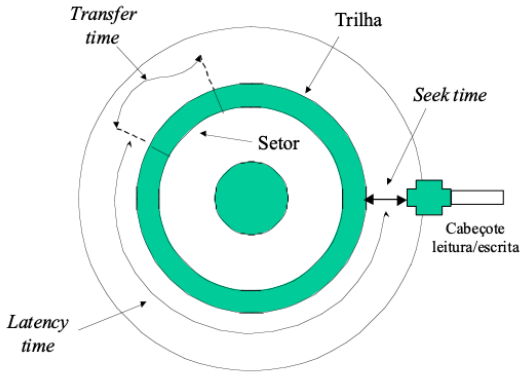


- Se visualizarmos a FAT e arquivos numa ferramenta de visualização binária, teríamos algo como mostrado abaixo:
-

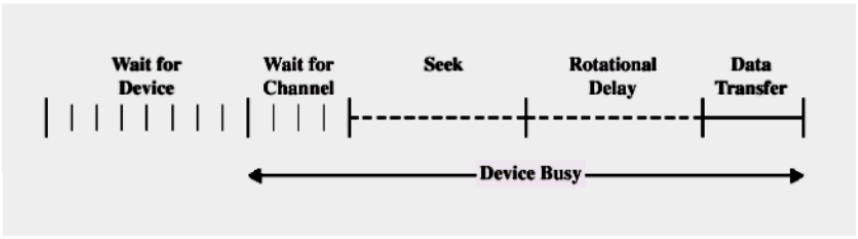


FATORES DE DESEMPENHO DE ACESSO A DISCOS

- Existem três fatores que alteram o desempenho de acesso aos dados de um disco:
  - **Tempo de busca (seek time)**: tempo necessário para posicionar o cabeçote de leitura/gravação na trilha
  - **Tempo de latência rotacional**: tempo necessário para atingir o setor a ser lido/escrito.
  - **Tempo de transferência**: tempo para escrita/leitura efetiva dos dados.

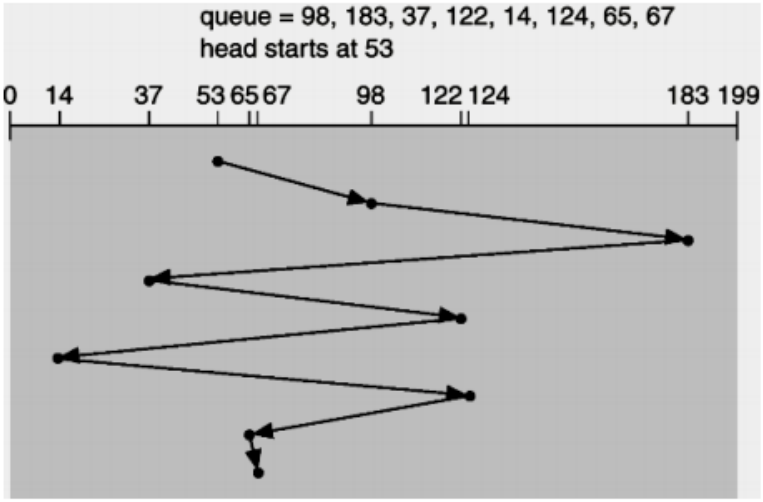


$$t_{acesso} = t_{seek} + t_{latência} + t_{trasnf}.$$



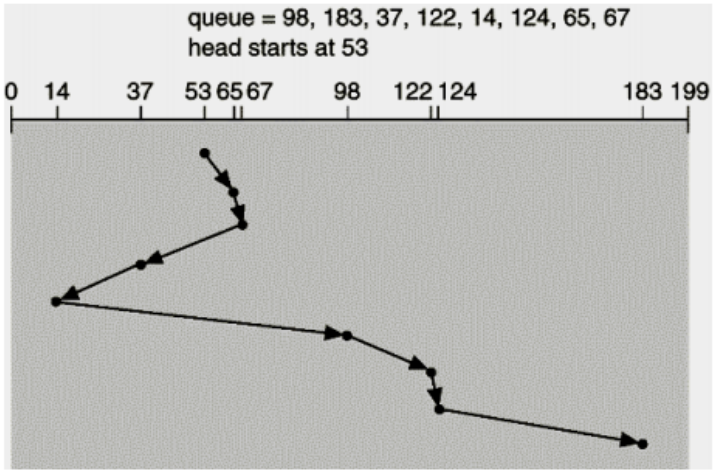
- Como o gerenciador de dispositivos recebe requisições de acesso ao disco de vários processos, ele precisa escalonar estas requisições. Existem diversas estratégias de escalonamento, de modo a tentar minimizar o número de movimentações da cabeça de leitura/gravação.

- **ALGORITMO FCFS(FIFO)** : acessa os blocos na ordem que as requisições são solicitadas.



Distância = 640 trilhas

- **ALGORITMO SSTF (SHORTEST SEEK TIME FIRST) :** seleciona a requisição com o menor tempo de seek em relação à posição atual do cabeçote de leitura/gravação.



EXERCÍCIO COM DISCUSSÃO EM DUPLAS

Explique por que, geralmente, o algoritmo SSTF é mais eficiente em termo d distância percorrida do que o FIFO.

EXERCÍCIOS EXTRA-CLASSE

1. No sistema operacional Linux também existe uma tabela de IRQs ? Caso afirmativo, como podemos consultá-la ?
2. Mostre como funciona o algoritmo SLTF para acesso a blocos de discos.
3. Mostre como funciona o algoritmo SPTF para acesso a blocos de discos.