Universidade Federal de Uberlândia
Faculdade de Computação
Sistemas Operacionais

Sistema de Arquivos

Interfaces

Prof. Dr. Marcelo Zanchetta do Nascimento

Roteiro

- Introdução
- Arquivos
- Atributos de Arquivos
- Extensão de Arquivos
- Operações de Arquivos
- Diretórios
- Leituras Sugeridas

Introdução

- Devido a ubiquidade dos computadores na nossa vida, conceitos como **memória** e **armazenamento** são tão comuns que raramente pensamos na diferença fundamental entre eles;

- **Memória:** É uma forma de conter informação, a qual é **não persistente**.

  - No contexto de S.O. é onde os processos e dados dos programas residem durante sua execução.

- **Armazenamento:** **Persistente** e geralmente uma ou mais ordens de grandeza maior que a memória.

- Necessária para reter a informação por períodos de tempo maiores que a vida de um processo.

Introdução

Persistência da Informação:

- Requisitos essenciais para armazenamento por longo prazo;
- Deve ser possível armazenar uma quantidade muito grande de informação;
- A informação deve sobreviver ao término do processo que a usa;
- Múltiplos processos têm de ser capazes de acessar a informação concorrentemente;

Sistemas de Arquivos

- As aplicações não devem se preocupar com o funcionamento interno dos dispositivos de armazenamento;
- Como identificar, organizar, acessar e proteger essas informações?
- **Uma das principais funções do S.O.** é abstrair a informação armazenada e torná-la facilmente acessível para os programas;
- O S.O. implementa esta abstração por meio do conceito de **arquivos**;
- O qual é tratado pelo S.O. como **Sistema de Arquivos**.

Sistemas de Arquivos

- Do ponto de vista do **Usuário**:
  - O que constitui um arquivo?
  - Como os arquivos são nomeados?
  - Como os arquivos são protegidos?
  - Quais operações são permitidas em arquivos?

Sistemas de Arquivos

- Do ponto de vista do **Sistema de Arquivos**:
  - Tipo de estrutura de dados usada para estruturar os bits de arquivos;
  - Quantos bits há em um setor?
  - Quantos setores há num bloco?
  - Como indexar a localização dos arquivos no disco?
  - Como alocar espaço para um novo arquivo?
  - Como manter controle do espaço disponível no disco?

Abstração – Arquivo

Arquivos:

- São unidades lógicas de dados criadas por processos;
- Em geral, um disco contém milhares de arquivos independentes um dos outros;
- É possível entender a abstração de arquivos como uma espécie de espaço de endereçamento;

Revisão sobre HDDs

- Inventados na década de 50;
- Originalmente chamados “fixed disks” ou “winchesters” (nome usado pela IBM);
- Posteriormente ficaram conhecidos como discos rígidos (HDD – Hard Disk Drive) para se distinguirem dos “floppy-disks”;
- Técnica de armazenamento magnético similar a utilizada em fitas magnéticas;
- Facilmente gravado e apagado e mantém a informação gravada por muitos anos;

Revisão sobre HDDs

- **Formatação de baixo nível** → estabelece as trilhas e setores no prato. Os pontos de início e fim de cada setor são escritos no setor;

- **Formatação de alto nível** → escreve as estruturas de armazenamento de arquivos (FAT – File Allocation Table);

Revisão sobre HDDs

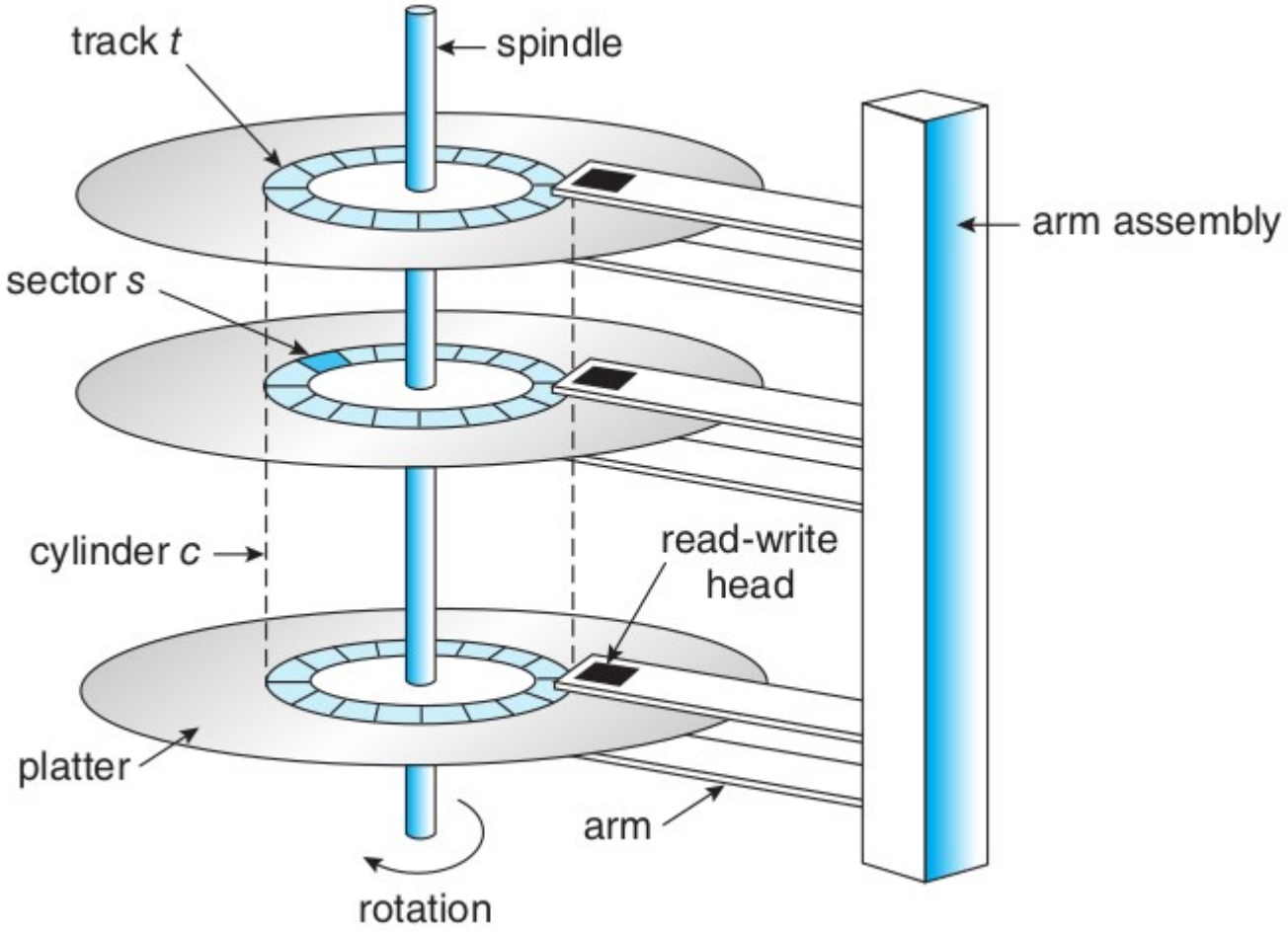


Figura 1 – Mecanismo empregado em um sistema HDD

SSDs: O que muda?

- Dispositivo de memória não volátil;

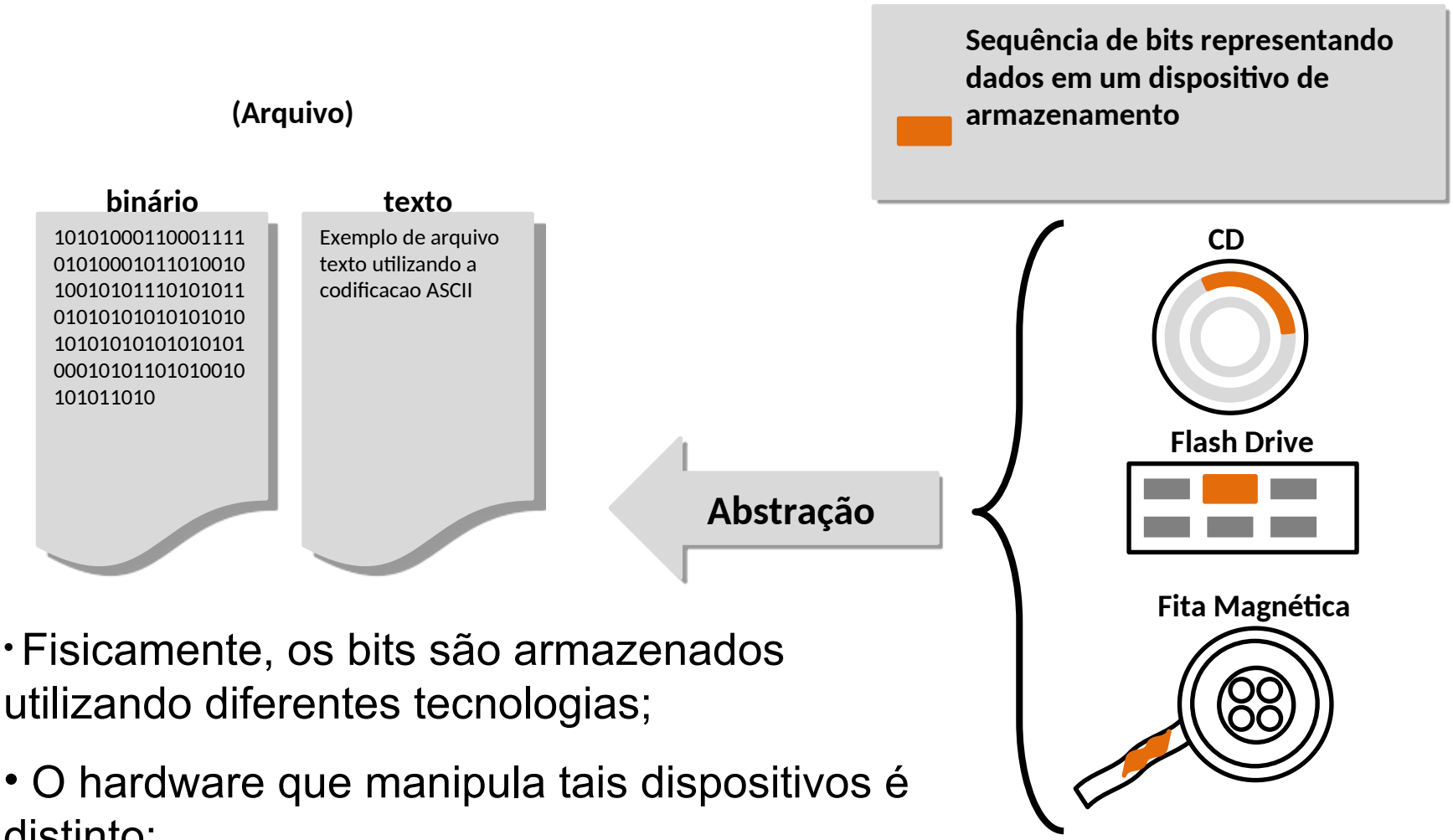
- Muitas diferenças: No entanto, a mais relevante é o fato de que SDDs não possuem partes móveis (Motores e braços móveis);

Figura 2 – Um circuito de um SSD de 3.5’’ .

SSDs

- Muito mais rápida que dispositivos HDD, no entanto, consideravelmente mais lenta que dispositivos de memória (meio termo);
- Vida útil menor que a dos principais dispositivos de armazenamento;
- **Por que os SDDs não substituem por completo os HDDs?**
- Os semicondutores se deterioram a cada ciclo de apagamento.

Abstração de Arquivos



- Fisicamente, os bits são armazenados utilizando diferentes tecnologias;
- O hardware que manipula tais dispositivos é distinto;
- Do ponto de vista do usuário/programador esses detalhes não importam.

Sistema de Arquivos

Sistema de arquivo organizado dentro de camadas

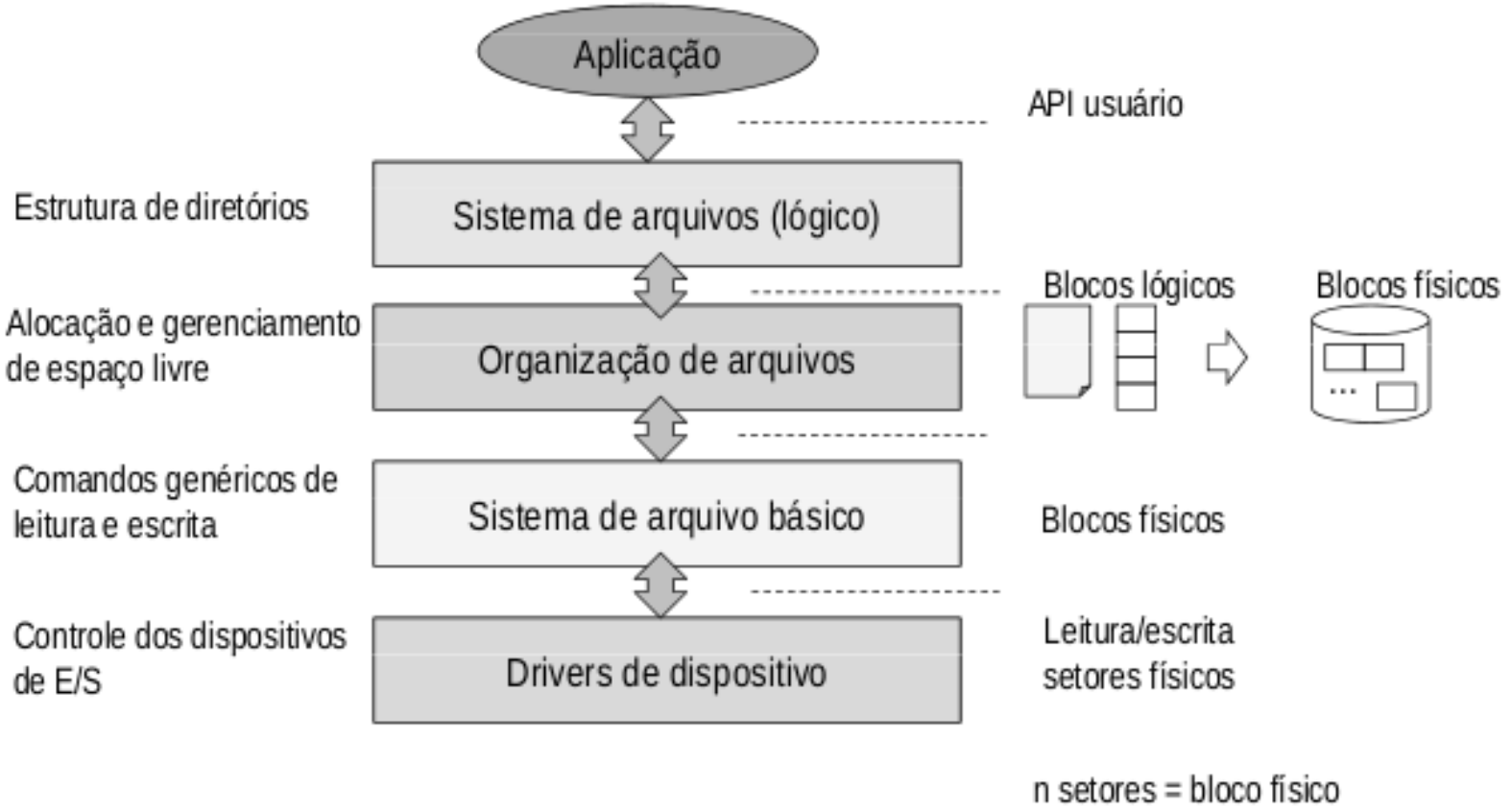


Figura 3 – Representação do Sistema de Arquivos em Camadas

Nome dos Arquivos

Nomeação de arquivos

- Quando arquivos são criados e são representados por nome simbólico de seus atributos;
- As regras extraídas do que constitui um nome válido variam de SO para SO;
- Muitos sistemas de arquivos permitem nomes com tamanhos de até 255 caracteres.
- Virtualmente todos os S.O.s permitem cadeias de caracteres de 1 a 8, sendo o primeiro uma letra e todos os demais letras ou números;
  - Exemplo: A-Z, a-z, 0-9, %, !,) e &.
- Sistemas também permitem uso de UNICODE;

Extensão dos Arquivos

Extensão:

- É um adendo ao nome do arquivo para
  - indicar o tipo de arquivo e os programas adequados para manipulá-los;
  - facilitar a organização e identificação do tipo de arquivo.
- Varia de sistema operacional para sistema operacional
  - Limitação da quantidade de caracteres usados no nome;
  - Caracteres permitidos;
  - *Case sensitive* ou não.

Extensão de Arquivos

Exemplos

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

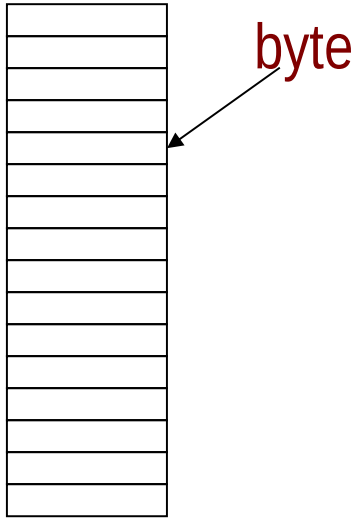
Tabela 4 – Exemplo de extensão de arquivos

Formato do Arquivo

- Muitos arquivos estruturam a informação de uma forma particular;
- Instruções sobre como a informação está estruturada no arquivo geralmente são codificadas dentro do próprio arquivo numa seção chamada **cabeçalho do arquivo**;
- Mas como é essa organização dentro do sistema de Arquivo do S.O.?

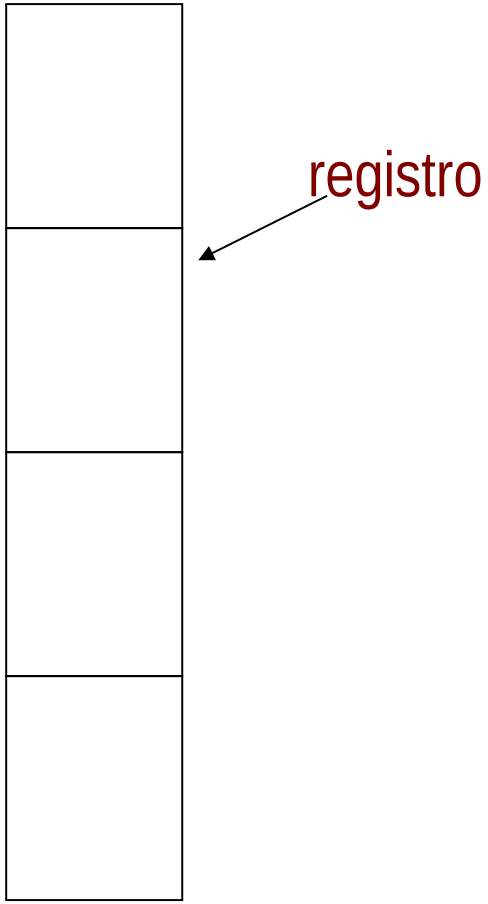
Estruturas de Arquivos

- Consiste em como os seus dados estão internamente armazenados;
- Arquivos podem ser estruturados de diferentes maneiras:
  - Sequência não estruturada de bytes
    - Para o SO, os arquivos são apenas conjuntos de bytes;
    - SO não se importa com o conteúdo do arquivo;
    - Significado deve ser atribuído pelos programas em nível de usuário (aplicativos);
    - Exemplo: UNIX e Windows;



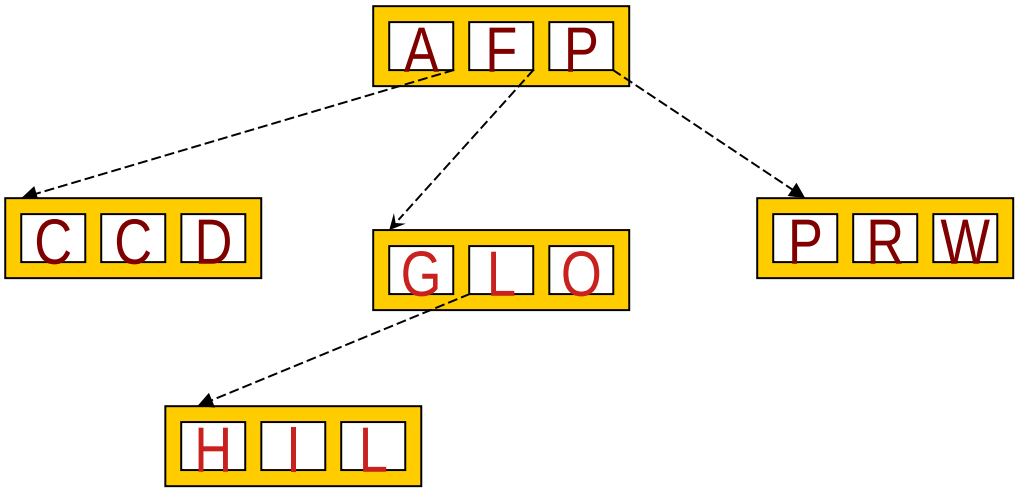
Estruturas de Arquivos

- Sequência de registros:
- Arquivo é “interpretado” como uma sequência de registros, isto é
- Tamanho fixo
- Estrutura interna
- Operações leem/escrevem registros
- Nenhum sistema usa esse modelo como sistema primário de arquivos
- Empregado em sistema de cartão perfurado



Estruturas de Arquivos

- Árvore de registros:
- Conjunto de registros não necessariamente de mesmo tamanho
- Possuem um campo de acesso (chave)
- Comum em *mainframes*
- Método ISAM (*Indexed Sequential Access Method*)



Tipos de Arquivos

- **Arquivos Regulares:** informação e programas;
  - arquivos de dados em ASCII e binário
- **Diretórios:** arquivos do sistema que mantêm a estrutura do sistema de arquivos;
- **Arquivos Especiais de Caracteres:** são relacionados a Entrada/Saída e usados para modelar dispositivos de Entrada/Saída, como terminais, impressoras, redes, etc;
- **Arquivos Especiais de Blocos:** são usados para modelar discos;

Tipos de Arquivos

executável
(*formato a.out*)

biblioteca

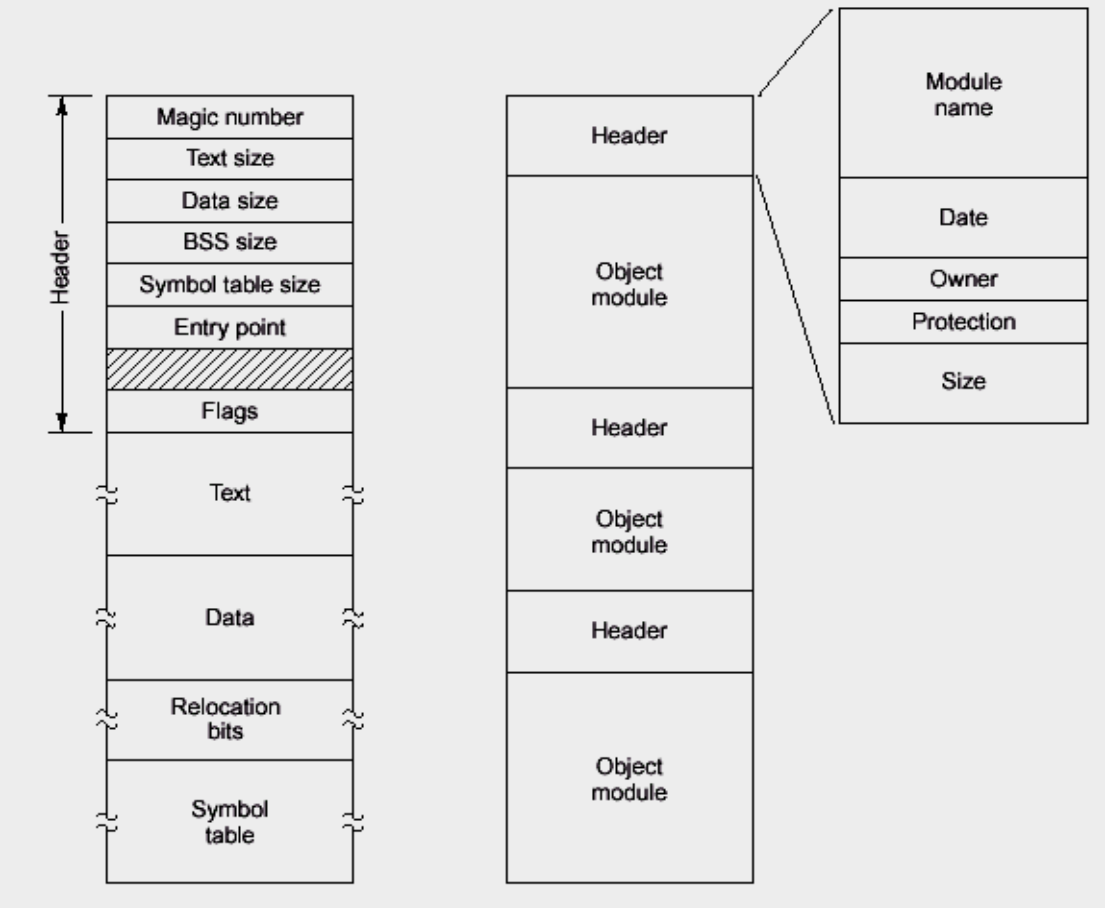


Figura 5 – Representação de Arquivo em Unix

Métodos de Acesso

- Forma pela qual o conteúdo de um arquivo é acessado
- Métodos elementares de acesso:
  - **Acesso sequencial:** Acesso a um arquivo é feito através de primitivas (chamadas de sistema) do tipo *read* e *write;*
    - Cada chamada de sistema *read* retorna ao processo os dados seguintes àqueles que foram lidos na chamada anterior.
  - **Acesso relativo:** Provê uma chamada de sistema específica para indicar o ponto em que um arquivo deve ser lido/escrito;
    - Implementado através da abstração de “posição corrente no arquivo”. Uma chamada pelo *seek*.

Atributos de Arquivos (Metadados)

- Mantido na estrutura de diretório
  - Cada arquivo possui uma entrada associada
- Nome: informação simbólica para referenciar o arquivo
- Tipo: binário, texto, executável, caracter e bloco
- Localização: posição em um determinado dispositivo E/S
- Tamanho: número de *bytes* que compõem o arquivo
- Proteção: controla acesso de leitura, escrita e execução
- Hora e data de criação, identificação do usuário: informações destinadas à proteção, segurança e monitoração
- Varia de sistema operacional a sistema operacional

Atributos de Arquivos

Atributo	Significado
Proteção	Quem tem acesso ao arquivo e de que modo
Senha	o arquivo requer uma senha para que possa ser acessado
Criador	ID do usuário que criou o arquivo
Proprietário	ID do usuário que detêm esta cópia particular do arquivo
Flag Leitura	Indica se o arquivo é apenas para leitura
Flag de Oculto	Indica se o arquivo é oculto
Flag de Sistema	Indica se o arquivo faz parte do sistema operacional
Flag de Arquivamento	Indica se o arquivo possui ou não backup
Flag ASCII/Binário	Indica o modo do arquivo ASCII ou binário
Flag Acesso Aleatório	Indica se o arquivo pode ser acessado de forma aleatória ou apenas sequencial
Flag de Temporário	Indica se o arquivo é temporário
Data Criação	Data em que o arquivo foi criado
Data Último Acesso	Data em que o arquivo foi acessado pela última vez
Data Última Alteração	Dada em que o arquivo foi alterado pela última vez
Tamanho Atual	Tamanho em bytes do arquivo

Operações com Arquivos

- Suportadas via chamadas do sistema;
- Operações com arquivos são tão pervasivas na utilização de um sistema computacional que as chamadas do sistema são mapeadas em programas do usuário e até mesmo embutidas diretamente no sistema de gerenciamento gráfico;
- Em Linux: linux-3.X.Y/include/linux/fs.h

1. criar
2. deletar
3. renomear
4. recuperar atributos
5. definir atributos
6. abrir
7. fechar
8. ler
9. escrever
10. acrescentar
11. procurar / reposicionar

Operações: Chamadas de Sistemas

```
#define BUF_SIZE 4096
#define OUTPUT_MODE 0700
int main(int argc, char *argv[]) {
  int in_fd, out_fd, rd_count, wt_count;
  char buffer[BUF_SIZE];
  if (argc!=3) exit(1);
  in_fd = open(argv[1], O_RDONLY);
  if (in_fd < 0) exit(2);
  out_fd = create(argv[2], OUTPUT_MODE);
  if (out_fd < 0) exit(3);
```

Operações: Chamadas de Sistemas

```
while((rd_count = read(in_fd, buffer, BUF_SIZE)) > 0)
{
    wt_count = write(out_fd, buffer, rd_count);
    if (wt_count <= 0) exit(4);
}
 close(in_fd);
 close(out_fd);
 if (rd_count == 0) exit(0);
 else exit(5);
}
```

Arquivos: Lista de Acesso

- Importante controlar o acesso aos arquivos devido a questões de segurança e de confidencialidade
- Objetivo é evitar acessos indevidos a arquivos
- Baseado na identificação dos usuários
- Sistema de autenticação padrão (*login name* + senha)
- Usuários possuem direitos de acessos
- Solução típica:
  - Lista de acesso e grupo

Exemplo: Lista de Acesso

Linux:

- Cada objeto oferece 3 *bits* (rwx) para três domínios diferentes: proprietário, grupo e outros
- Problema de flexibilidade
  - Quando um usuário pertence a vários grupos ele é identificado por um grupo primário e o arquivo (/etc/groups) mantém todos os grupos a que ele pertence

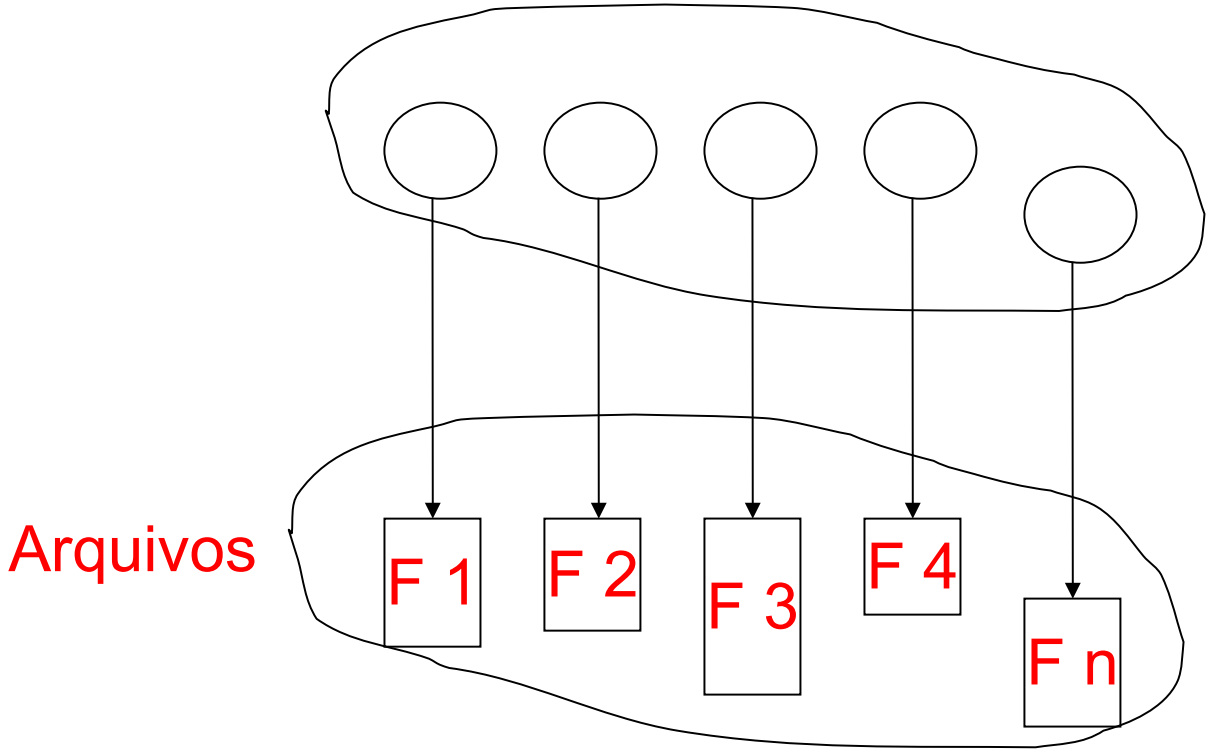
r w x r- - r - - 1 mary staff 214056 May 30 22:19 windbind.pdf

Estrutura de Diretório: Conceitos

- O diretório pode ser visto como uma tabela de símbolos, que traduz nomes de arquivo nas entradas de diretório;
- As operações em um diretório:
  - Procurar por um arquivo:procurar a entrada para um determinado arquivo
  - Criar um arquivo:
  - Excluir um arquivo:
  - Listar um diretório: listar os arquivos
  - Renomear um arquivo:
  - Atravessar o sistema de arquivo: acessar cada diretório e cada arquivo dentro de uma estrutura de diretório.

Estrutura de Diretório

- Uma coleção de nodes contendo informações sobre todos os arquivos



Ambos as estruturas de arquivos e diretórios residem no disco

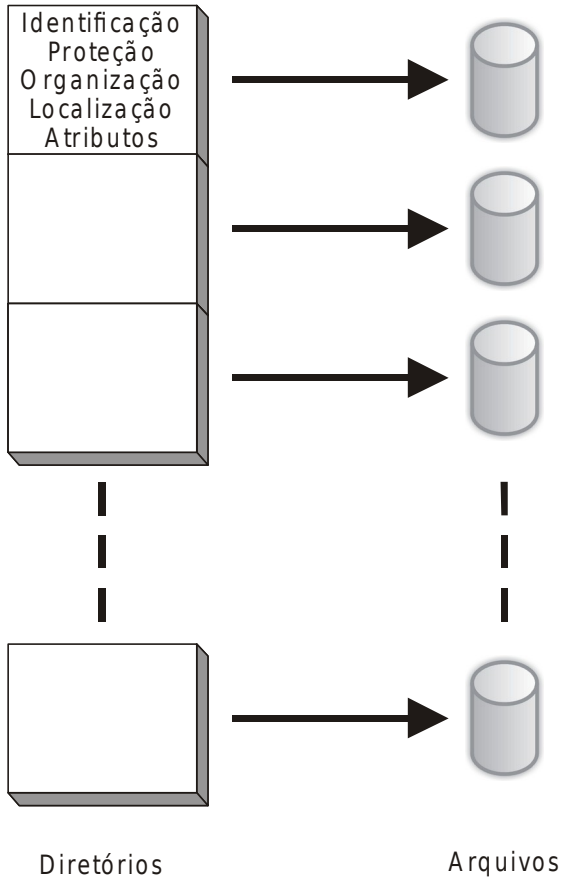
Estrutura de Diretório

Exemplos de operações com diretório (UNIX):

- **`Create;`**
- **`Delete;`**
- **`Opendir;`**
- **`Closedir;`**
- **`Readdir;`**
- **`Rename;`**

Hierarquia de Diretório

- Estrutura de diretórios de nível único
  - Usado em sistemas antigos;
  - Todos os arquivos existiam em um único diretório;
  - Apenas um diretório contém todos os arquivos : diretório raiz (*root directory);*
  - **Desvantagem:**
    - Sistemas com vários usuários podem criar arquivos como mesmo nome;

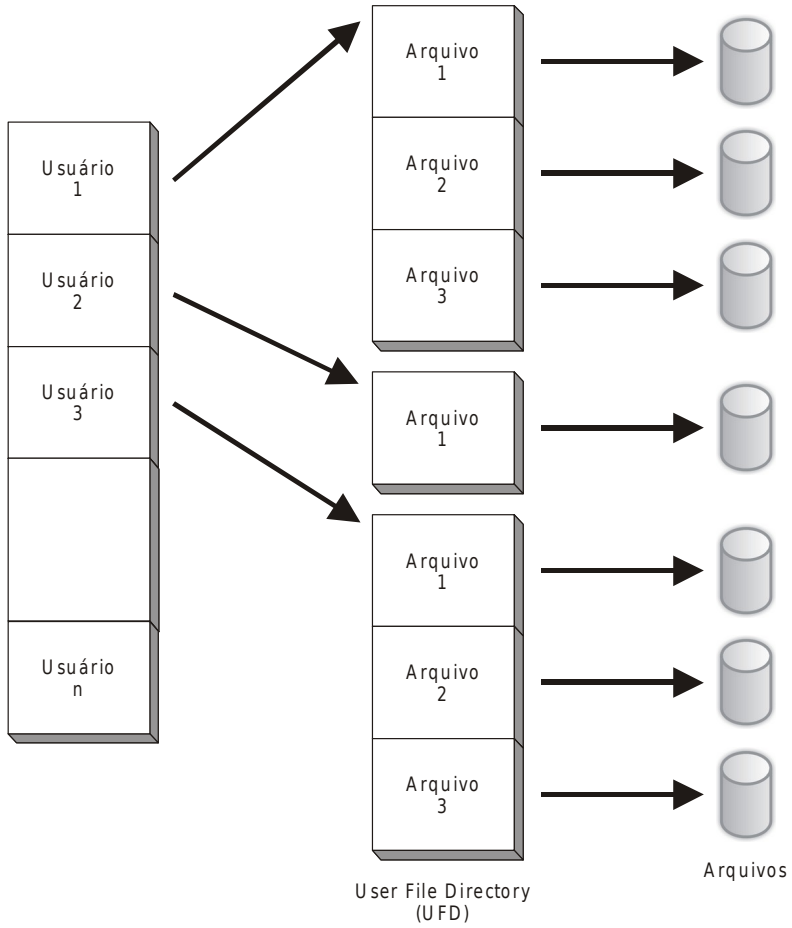


Hierarquia de Diretório

- Estrutura de diretórios com dois níveis

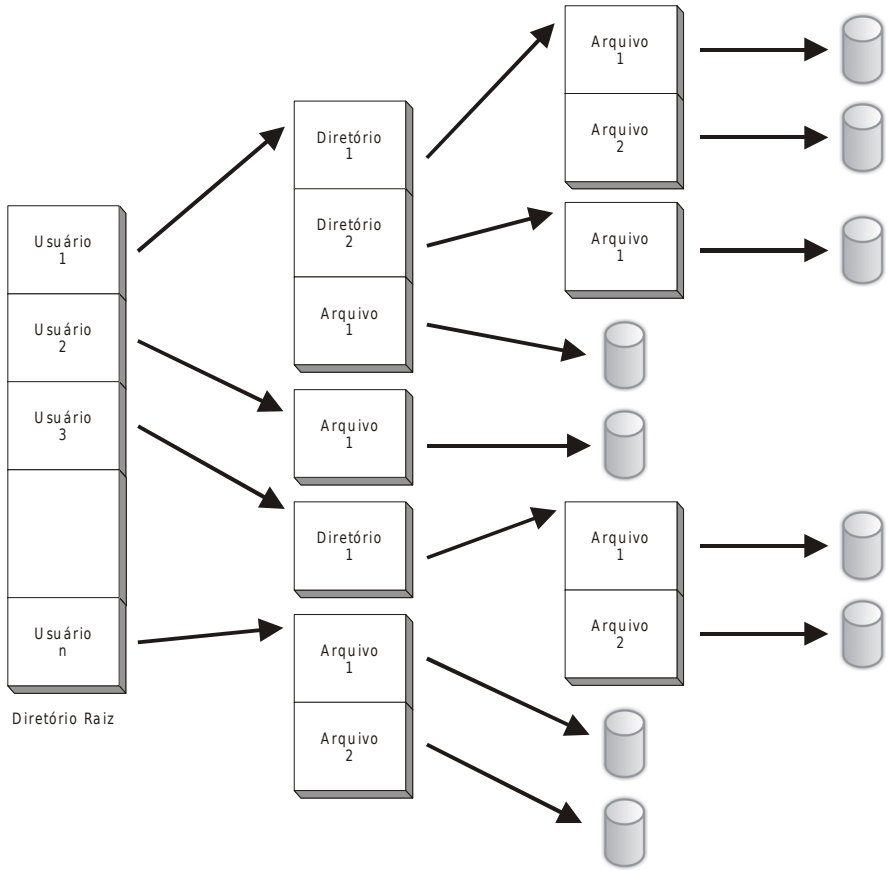
- Cada usuário possui um diretório privado;

- Sem conflitos de nomes de arquivos;



Hierarquia de Diretório

- Estrutura de diretórios em Árvore:
  - Possui um diretório raiz (*master*)
  - Usuários podem criar diversos diretórios que agrupam arquivos;
  - Busca eficiente;
  - Capacidade de agrupamento;
  - Diretório corrente (diretório de trabalho)
    - cd /spell/mail/prog



Nomes de Caminhos

- Forma de especificar onde na árvore de diretório encontra-se um dado arquivo;
- **Diretório de Trabalho** → usualmente o caminho na árvore de diretório onde encontra-se o arquivo em execução;
- **Diretório HOME** → diretório associado pelo SO a conta do usuário;
- Duas formas de especificar um caminho:
  - **Caminhos absolutos** → todos os subdiretórios desde o diretório raiz;
  - **Caminhos relativos** → subdiretórios a partir do diretório de trabalho;
- "•" → diretório atual;
- "••" → diretório anterior;

Nomes de Caminhos

WINDOWS

```
C:\tmp\dummy.txt
```

UNIX

```
/tmp/dummy.txt
```

MULTICS

```
>tmp>dummy.txt
```

Operações em Diretório

- **CREATE** → cria um diretório vazio. Contém “.” e “..”;
- **DELETE** → remove um diretório. Apenas diretórios vazios podem ser removidos;
- **OPENDIR** → permite a leitura de um diretório. Funciona analogamente a abertura de um arquivo para leitura;
- **CLOSEDIR** → quando acabar de ser lido, o diretório deve ser fechado para liberar espaço na tabela interna;
- **READDIR** → devolve a próxima entrada em um diretório aberto;
- **RENAME** → em muitos aspectos, os diretórios são como arquivos e podem ter seu nome alterado;

Operações em Diretório

- **LINK** → a ligação (linking) é uma técnica que possibilita um arquivo aparecer em mais de um diretório;

  - Essa chamada de sistema especifica um arquivo existente e um nome de caminho e, então, cria uma ligação do arquivo existente com o nome especificado pelo caminho. Também chamado de (**hard link**)

- **UNLINK** → remove uma entrada de diretório. Se o arquivo sendo desligado estiver presente em apenas um diretório (não for um link), ele será removido do sistema de arquivos.

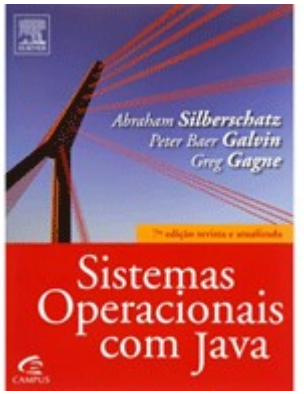
Exemplo de Diretório: UNIX

-rw-rw-r--	1 pbg	staff	31200	Sep 3 08:30	intro.ps
drwx------	5 pbg	staff	512	Jul 8 09.33	private/
drwxrwxr-x	2 pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2 pbg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1 pbg	staff	9423	Feb 24 2003	program.c
-rwxr-xr-x	1 pbg	staff	20471	Feb 24 2003	program
drwx--x--x	4 pbg	faculty	512	Jul 31 10:31	lib/
drwx------	3 pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3 pbg	staff	512	Jul 8 09:35	test/

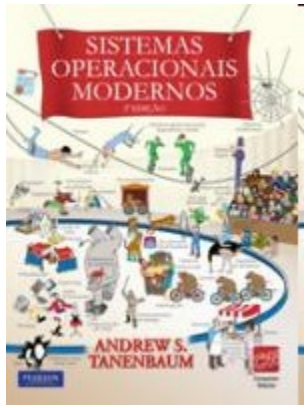
Figura 3 – Exemplo de Arquivos e Diretórios em Ambiente Unix

Leituras Sugeridas

- Silberschatz, A., Galvin, P. B. Gagne, G. Sistemas Operacionais com Java. 7º edição. Editora Campus, 2008.
  - Capítulo 10



- TANENBAUM, A. Sistemas Operacionais Modernos. Rio de Janeiro: Pearson, 3 ed. 2010.
  - Capítulo 4



Agradecimento

- Ao Prof. Dr. Daniel Abdala pelas notas de aula.