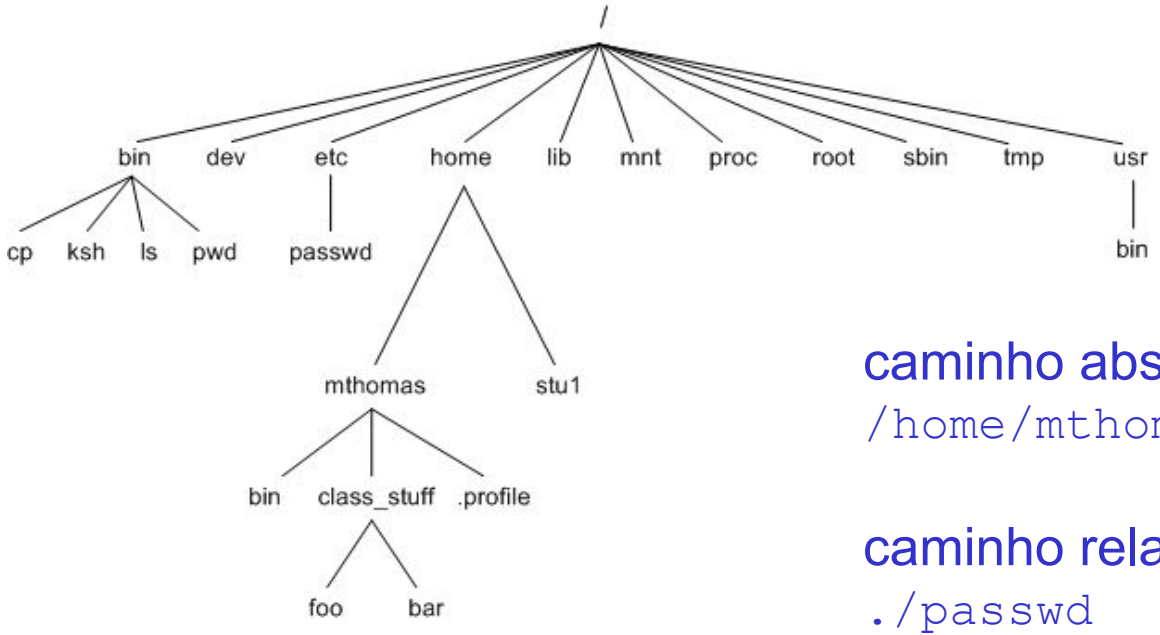
Sistema de Arquivos Unix

Hierárquico
- Estrutura em árvore
- Diretórios de arquivos são os nós internos
- Arquivos são as folhas
- Sem restrições quanto à largura e profundidade da árvore

Consistente
- Quase todos os objetos do sistema são representados como arquivos e podem ser utilizados através de uma interface de programação consistente e uniforme (p.ex. Para arquivos,diretórios,named pipes, dispositivos)
- Tratamento sintático igual a todos os tipos de objetos.
- Assim, as aplicações ficam mais independentes do tipo de mídia usado (e.g. HD, CD-Rom, SSD)

Simples
- A API oferece apenas algumas poucas - e versáteis - operações sobre arquivos
- Um sistema de arquivos lógicos pode consistir em multiplos sistemas de arquivos físicos
- Um sistema de arquivos pode ser ligado a qualquer caminho do sistema de arquivos virtual usando o comando mount.

Estrutura hierárquica do Sist. de Arquivos UNIX



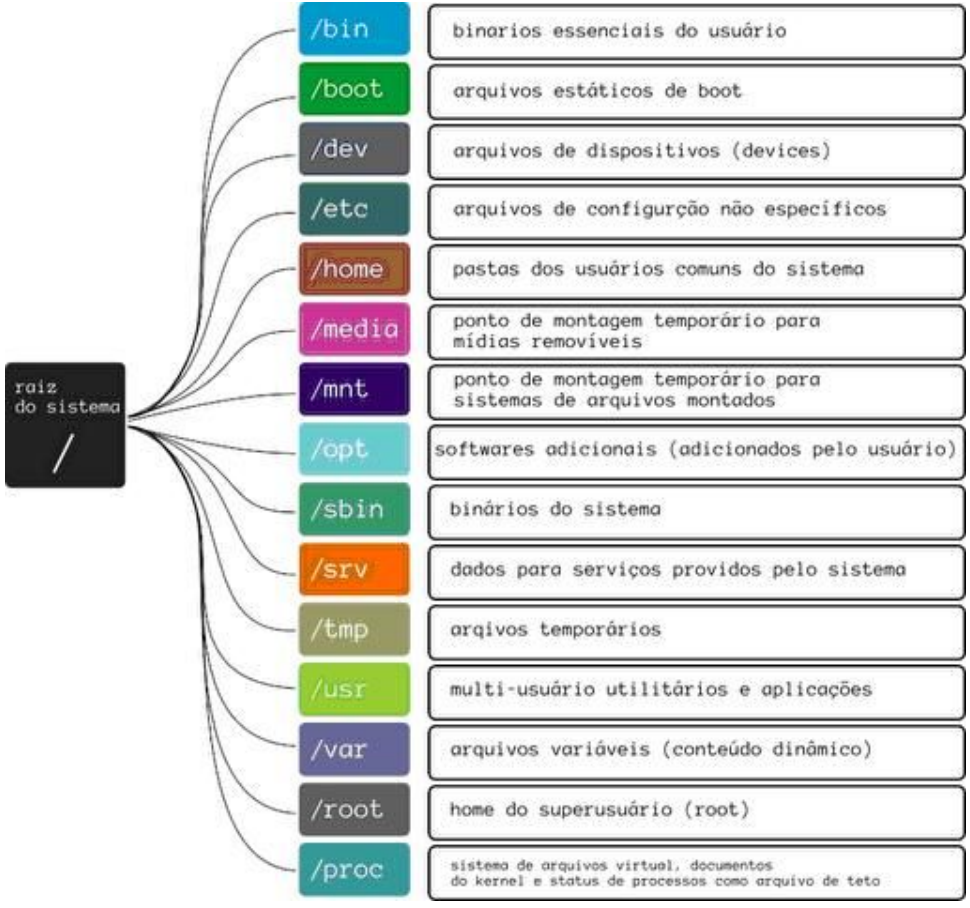
caminho absoluto: (path)

`/home/mthomas/profile`

caminho relativo:

`./passwd  (em /etc)`

Sistema de Arquivos Lógico do Unix



Tipos de Arquivo em UNIX

Do ponto de vista conceitual lógico, em Unix **tudo é arquivo**.

Exemplo de alguns tipos...

- **Arquivo Regular** – ASCII (data/ text) ou binário (p.ex. executável ou dados cifrados)
- **Diretório** –para agrupar arquivos e subdiretórios
- **Arquivos especiais** - representam dispositivos físicos usados para E/S
  - Dispositivos orientados a caractere - abstração de E/S serial (terminal, impressoras, sockets)
  - Dispositivos orientados a bloco – abstração de E/S de disco (HD, CD-ROM, DVD, etc.)
- **Link** – conexão entre um nome de arquivo e um i-node de outro arquivo, (i-node é a representação interna de um arquivo)
- **Named Pipe** – permite uma comunicação direta entre processos,

(Padrões de) Sistemas de Arquivos

Os sistemas de arquivo (FAT, NTFS, EXT2, ReiserFS) determinam como arquivos serão gravados fisicamente no disco rígido. Cada sistema de arquivos oferece algumas vantagens e desvantagens principalmente com relação a integridade dos dados, velocidade de acesso etc.

No Linux os principais sistemas de arquivos são (EXT2, EXT3, ReiserFS)

O sistema de arquivos second extended Filesystem (EXT2) foi desenhado como uma extensão de extended Filesystem (ext). O EXT2 oferece a melhor performance (em termos de velocidade e uso da CPU) entre todos os sistemas de arquivos suportados pelo Linux.

O sistema de arquivos EXT3 é uma evolução do sistema de arquivos EXT2, que foi originalmente desenvolvido por Stephen Tweedie, Rémy Card e Theodore Ts'o e outros.

Sabe-se que o sistema EXT3 possui excelente performance no gerenciamento de dados, tanto no que diz respeito ao armazenamento, quanto nas alocações e atualizações de informações.

EXT3 passou a ser integrado definitivamente ao Linux (kernel) a partir da versão 2.4

Arquivo (Regular) no Unix

String de Bytes

Não possui nenhuma propriedade predefinida

Formato e conteúdo é definido pelo usuário/programa criador

Restrito a um único mídia lógica

Protegido por controle de acesso:

- – r (read)
- – w(write)
- – x(execute)

Definidos para usuário, grupo e outros

Internamente, cada arquivo é representado por um inode.

7 tipos de arquivo:
Regular file,
Directory file,
Link file,
Character special file,
Block special file,
Socket file,
Named pipe file

Inode (index node)

Cada arquivo é representado por um i-node, uma estrutura que contém, entre outros:

/> Owner (UID,GID)

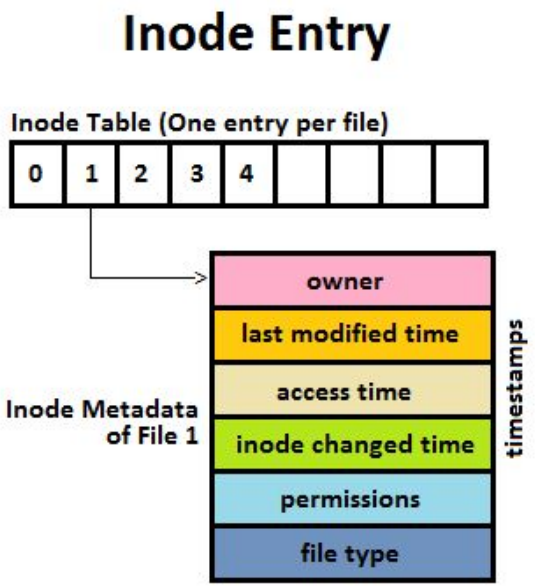
/> Permissões de acesso ao arquivo

/> Momento da última modificação e acesso

/> Tamanho do Arquivo

/> Tipo do arquivo (e.g. diretório, dispositivo de E/S, pipe, etc.)

/> Ponteiros para os blocos de dados no disco, com o conteúdo do arquivo



Fonte: https://web.cs.ucla.edu/classes/winter16/cs111/scribe/12d/index.html

Diretórios

São tratados como arquivos normais (apenas são marcados no inode como sendo um "diretórios"Diretório'

Uma entrada no diretório contém:

- Comprimento da entrada
- Nome (tamanho variável, até 255 caracteres)
- Inode number

Usuários identificam um arquivo através do seu pathname

Estes são mapeados pelo sistema de arquivo para numeros do inode

Se um path (caminho) começa com o"/",'é um nome absoluto e é resolvido desde o diretório raiz. it'is'absolute'and'is'resolved'up'from' the'root'directory' • Senão, o caminho é resolvido em relação o curetório corrente do usuário.

Link Simbolicos

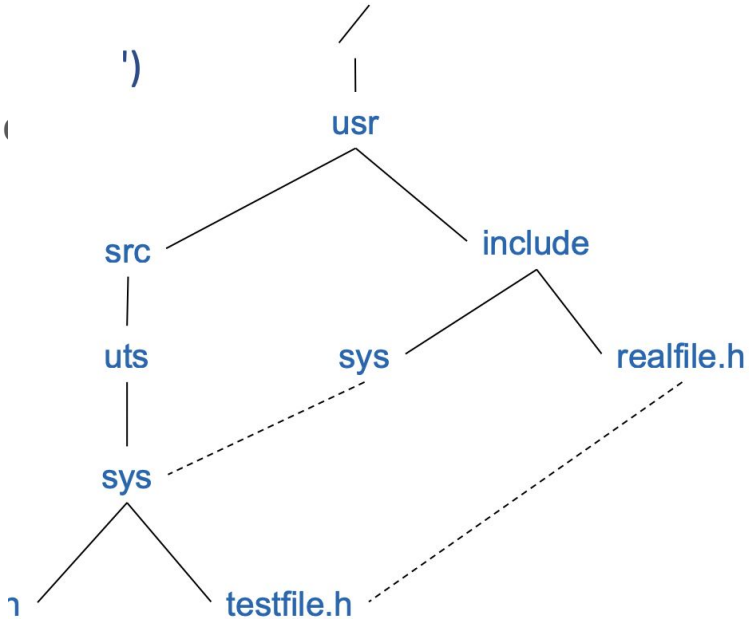
Em Unix pode-se usar links simbólicos para referenciar arquivos e diretórios através de diferentes caminhos -

symlink(nome_atual, nome_novo) - cria um caminho adicional para o recurso

Após executar

```
symlink("/usr/src/uts/sys","/usr/include/sys") (
symlink("/usr/include/realfile.h",'
"/usr/src/uts/sys/tesfile.h")
```

Temos 3 caminhos para o mesmo arquivo



Hard e Soft Link

Hard link

- Cria-se uma nova entrada de diretório que aponta para o arquivo referenciado
- Os hard links apontam para o mesmo i-node e são registrados pelo link counter do i-node do arquivo.
- Apenas quando o link counter chega a 0 o arquivo pode de fato ser removido.
- Esses links só podem ser criados para arquivos do mesmo sistema de arquivo lógico

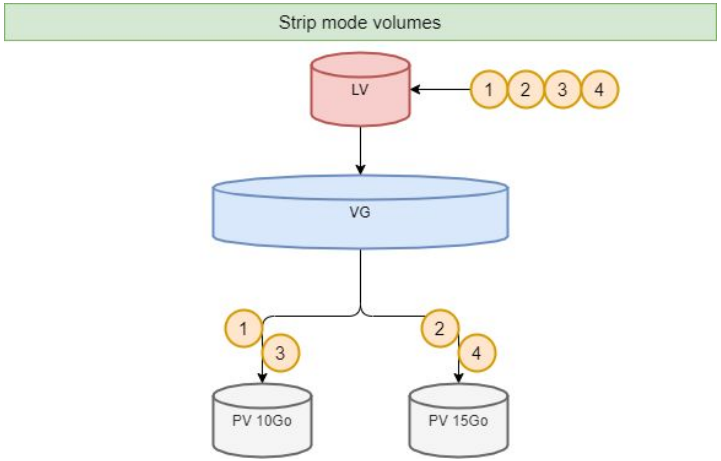
Soft link

- É um arquivo que contém o caminho para o arquivo ou diretório referenciado
- Esses links são interpretados e resolvidos em cada acesso
- Se o arquivo/diretório referenciado é removido, então o link passa a ficar inválido mas continua existindo

Sistema de Arquivo lógico vs físico

- Um sistema de arquivos lógico pode consistir em vários sistemas de arquivos físicos

- Um sistema de arquivos pode ser ligado a qualquer caminho da árvore do sistema de arquivos virtual através do comando "mount".

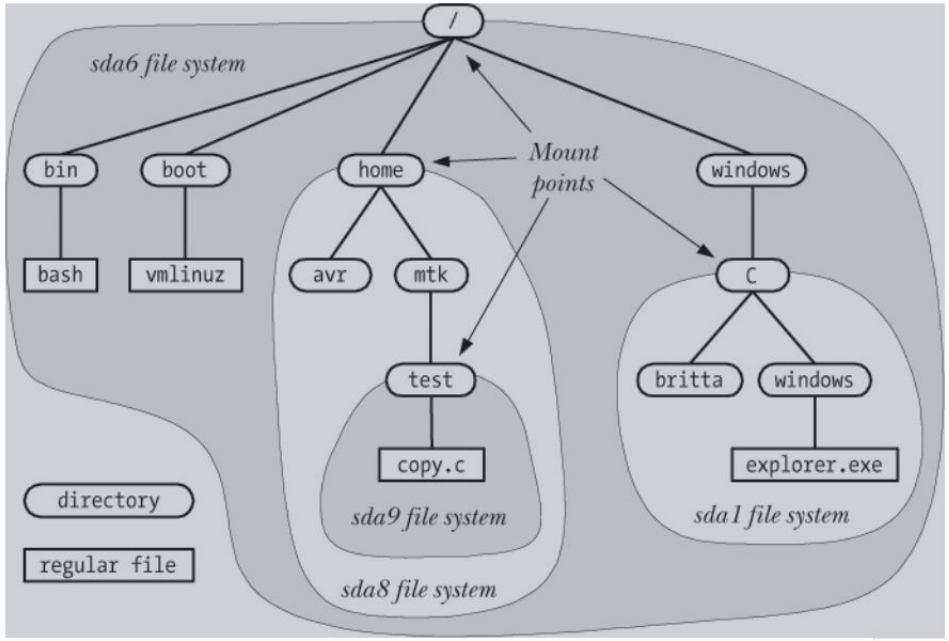


O comando mount

- Um sistema de arquivos pode ser ligado a qualquer caminho da árvore do sistema de arquivos virtual através do comando "mount".

- Os sistemas de arquivos montados são geridos pelo SO numa "tabela de montagem" que liga os caminhos aos mount points

- Isto permite identificar os inodes raiz dos sistemas de arquivos montados

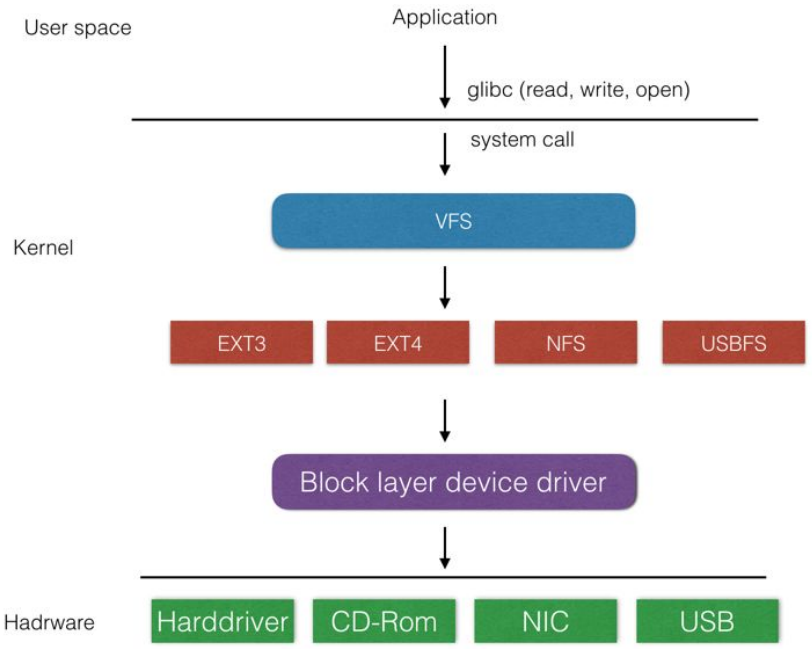


Virtual File System

VFS implementa uma interface genérica entre a implementações concretas de sistemas de arquivos (implementadas no núcleo) e as aplicações que acessam os arquivos

Assim provê interoperabilidade transparente entre sistemas de arquivo específicos.

Aplicações acessam diferentes Sistemas de arquivos em diferentes mídias usando uma interface de programação homogênea

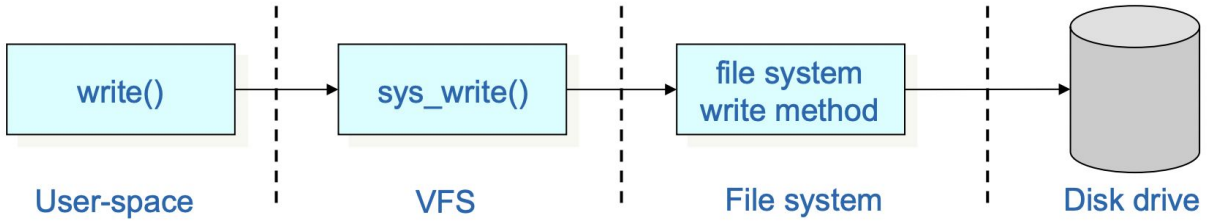


Exemplo: `write(f, &buf, len)`;

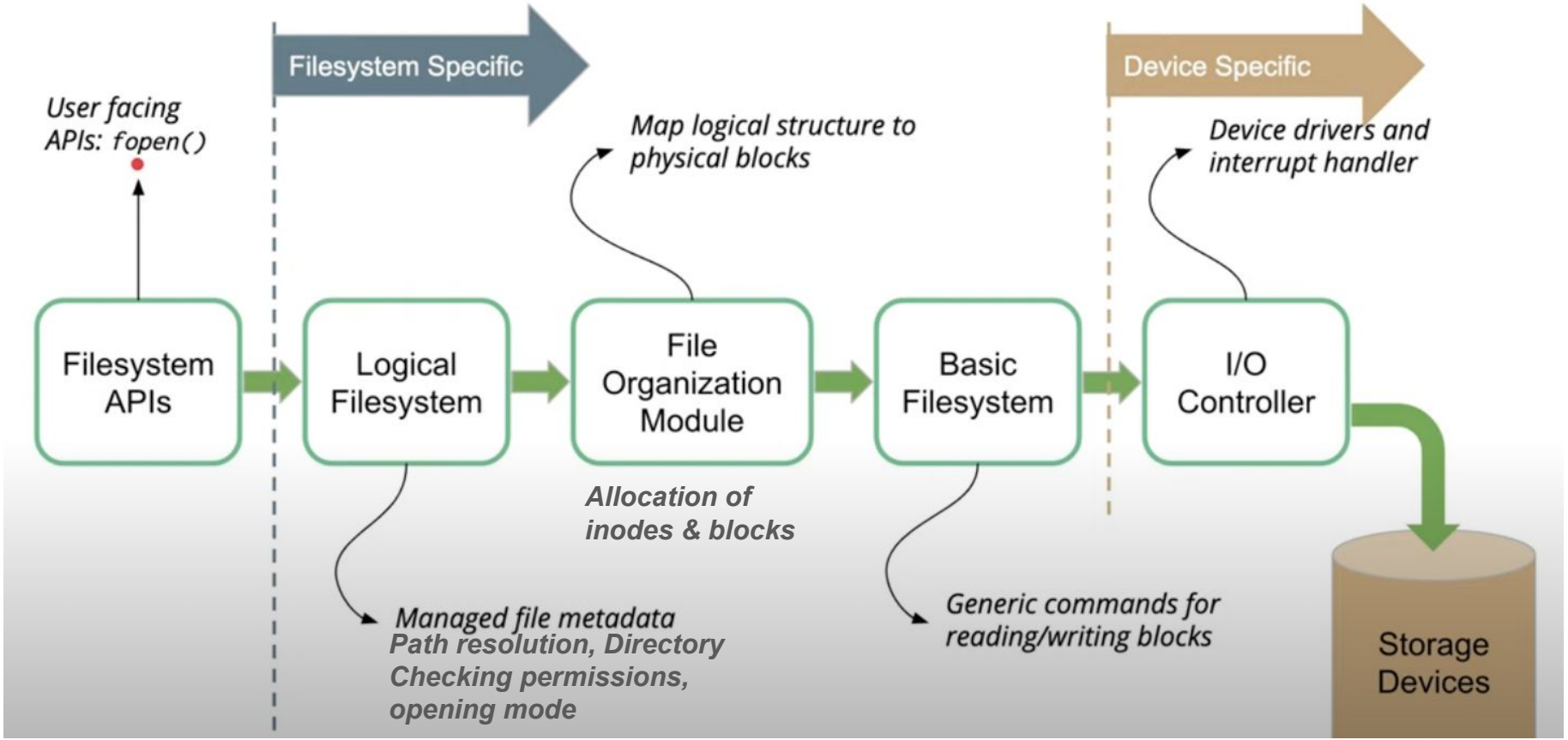
Essa chamada de função é traduzida para uma chamada de sistema de outro SA específico

E a system call é convertida para a implementação do sistema de arquivos concreto

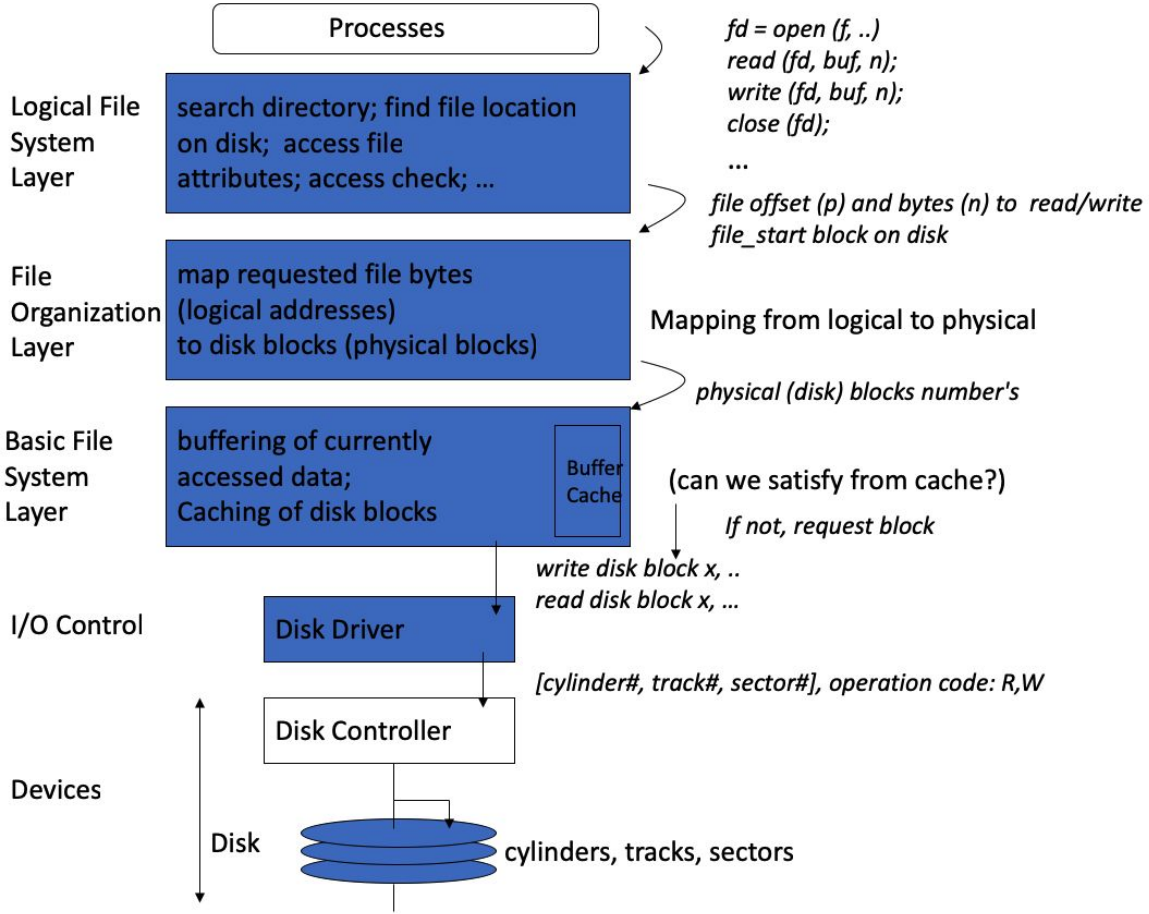
O sistema de arquivo executa o comando write()



Layered File System Organization



Camadas do Sistema de Arquivos



Arquivos estão em Blocos no Disco

Assim como em paginação, é interessante ter uma unidade de alocação de espaço no disco (o “bloco de disco”) de tamanho fixo.

Permitindo que arquivos possam crescer e ocupem blocos não consecutivos no disco.

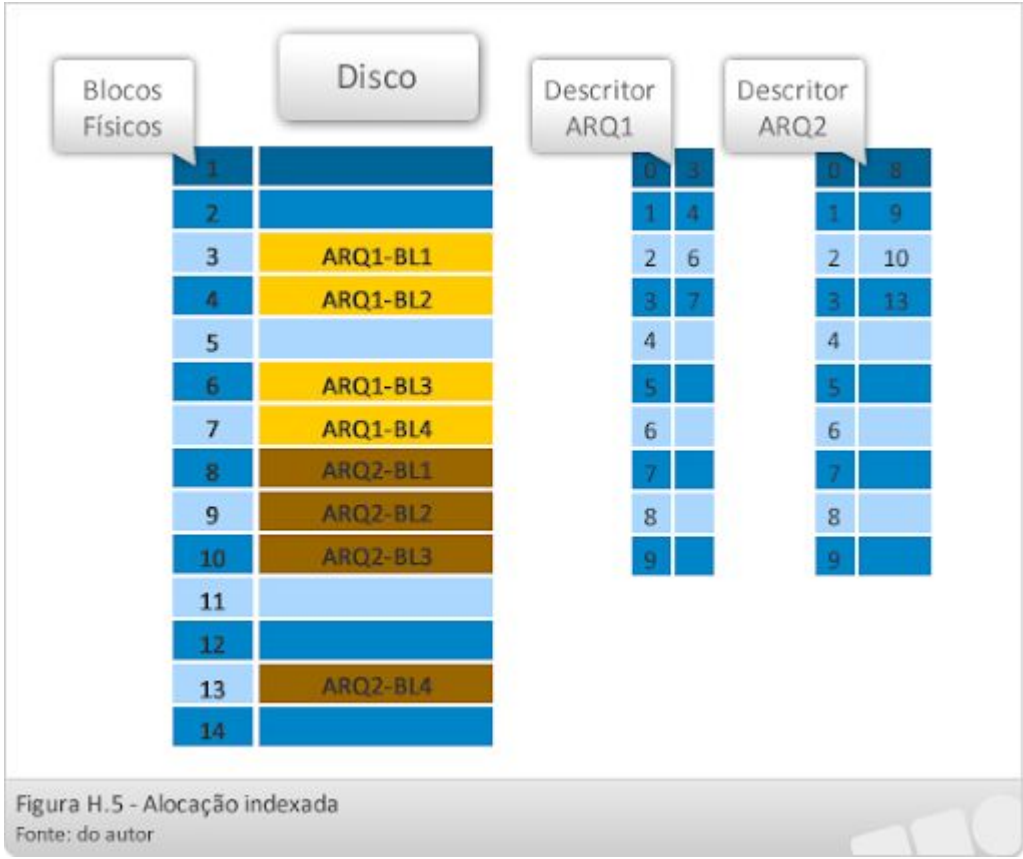
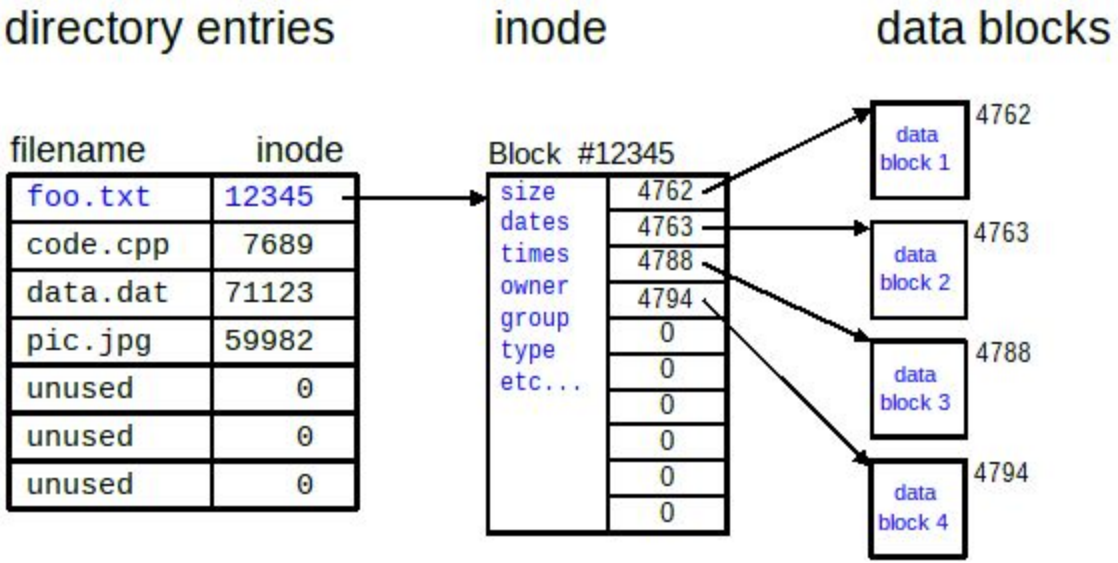


Figura H.5 - Alocação indexada

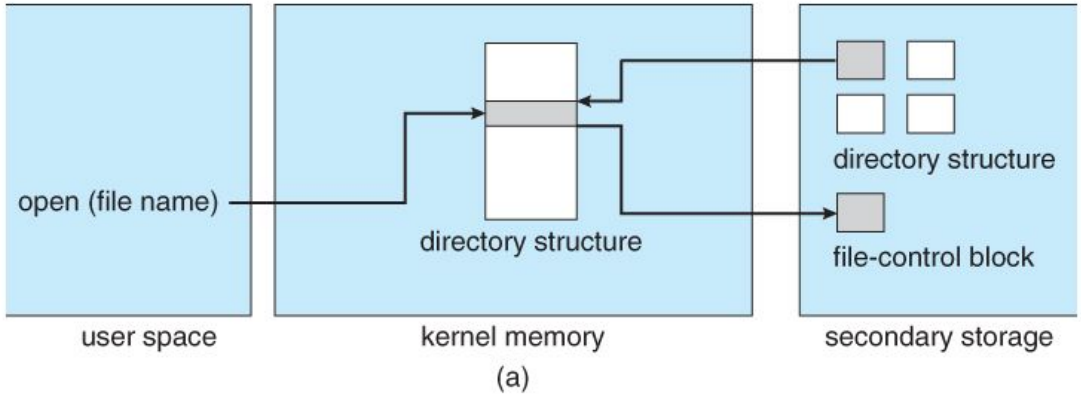
Fonte: do autor

Diretórios, i-nodes e blocos de dados



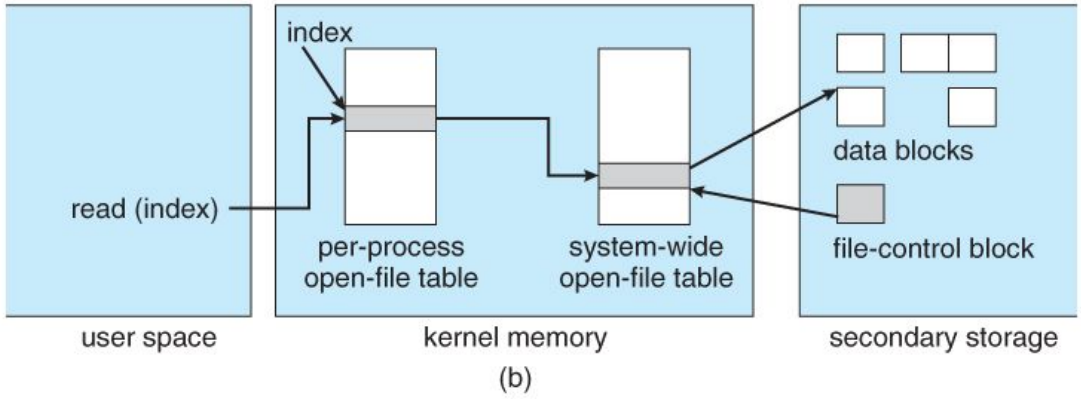
Estruturas em Memória Principal e Secundária

open() mapeia o path do arquivo ao seu inode



read(fd,...). fd é o indice para a per-process-OFT, que por sua vez referencia uma entrada no OFT do sistema,

Aqui se encontra o ponteiro para o inode do arquivo

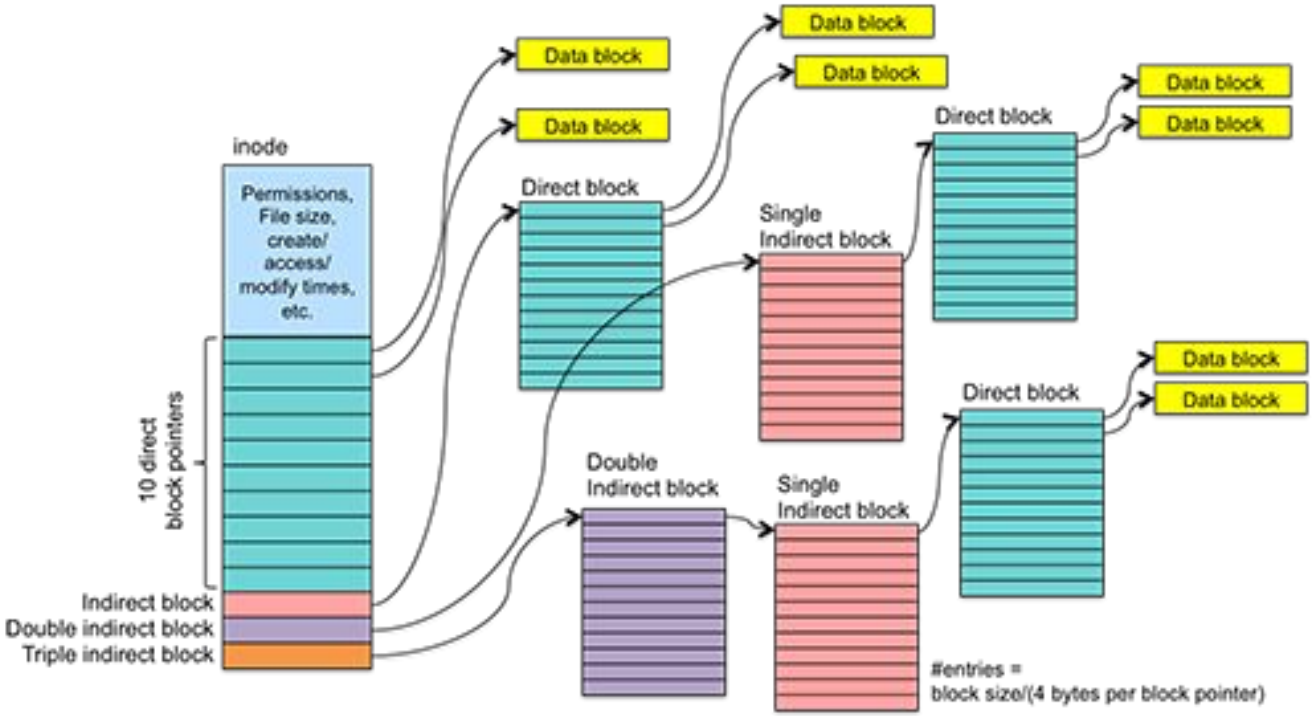


Inode, e diferentes tipos de blocos de índice

Usando…

- Direct index block),
- Single indirect block,
- Double indirect block e
- Triple indirect block

Inodes conseguem indexar bem tanto arquivos pequenos como muito grandes



Virtual File System

- Define operações uniformes para a interface de system call
- Faz a conversão para as estruturas de dados e operações de cada SA individual
- Implementa e gerencia os caches de inodes e entradas de diretório (dentries)