

Sistemas Operacionais

Sistema de Arquivos
Implementação



Visão do usuário

- Como arquivos são nomeados?
- Que operações são permitidas?
- Como o diretório é implementado?

Visão do implementador

- Como arquivos e diretórios são armazenados?
- Como o espaço em disco é gerenciado?
- Como tornar o sistema eficiente e confiável?

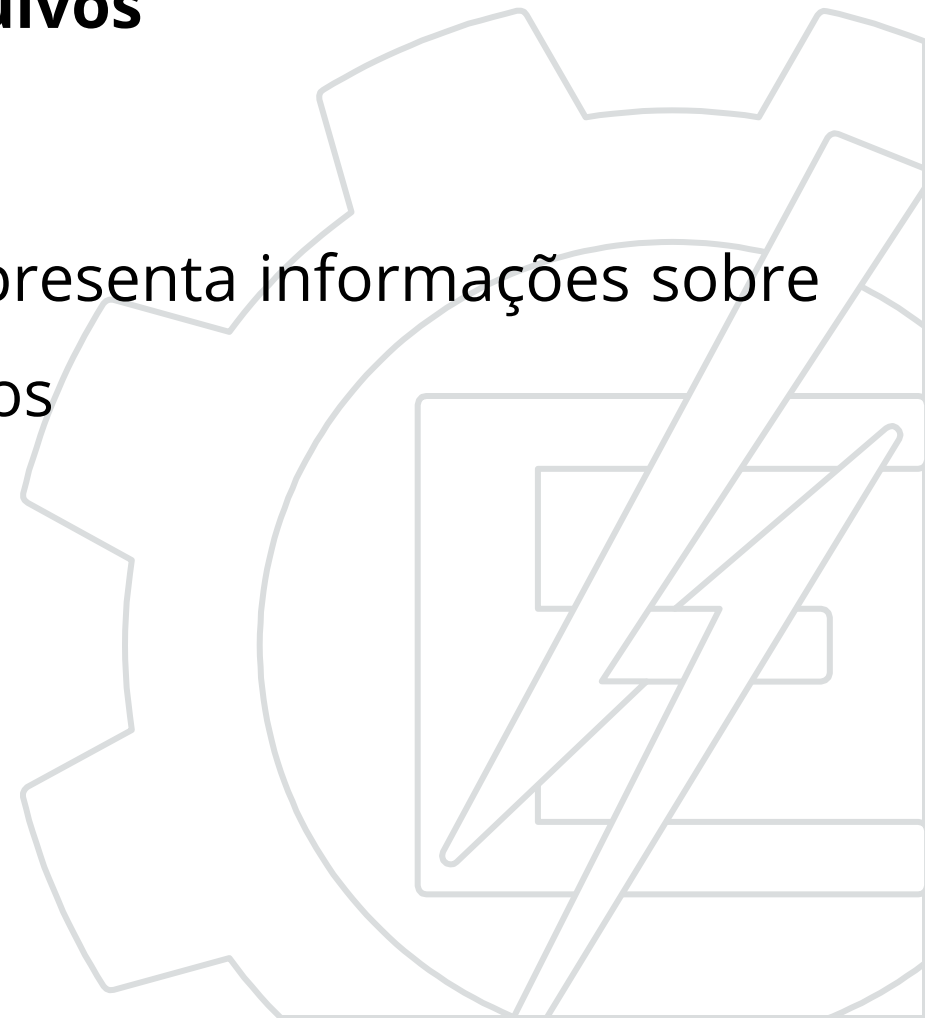


Estrutura de armazenamento - Sistemas de arquivos

- ***tmpfs*** — um sistema de arquivos “temporário” que é criado em memória principal volátil e tem seu conteúdo apagado quando o sistema é reinicializado ou cai.
- ***objfs*** — um sistema de arquivos “virtual” (essencialmente uma interface para o *kernel* que se parece com um sistema de arquivos) que fornece aos depuradores acesso a símbolos do kernel
- ***ctfs*** — um sistema de arquivos virtual que mantém informações de “contrato” para gerenciar que processos são iniciados quando o sistema é inicializado e devem continuar a ser executados durante a operação

Estrutura de armazenamento - Sistemas de arquivos

- ***procfs*** — um sistema de arquivos virtual que apresenta informações sobre todos os processos como um sistema de arquivos
- ***ufs, zfs*** — sistemas de arquivos de uso geral



- Existem dois tipos de sistemas de arquivos:
 1. Sistemas de disco (funciona em uma unidade de armazenamento); e
 2. Sistema de arquivo em rede (armazenamento é feito em outro computador da rede).
- O **NTFS** é um sistema de arquivos de disco que utiliza estrutura de Árvore B+ e que foi desenhada para minimizar a fragmentação.
- O NTFS permite 1) nome alternativo; 2) cota de armazenamento; 3) pontos de montagem; 4) ponto de junção; 5) ligação entre arquivos; 5) gerenciamento hierárquico do armazenamento; 6) controle de versão dos arquivos (via *shadow copy*); 7) compressão de arquivos; 8) armazenagem unificada de arquivos (múltiplas cópias de um mesmo arquivo são armazenadas uma única vez com referências múltiplas na tabela mestre); e 9) criptografia.

Sistema de Arquivos

Implementação

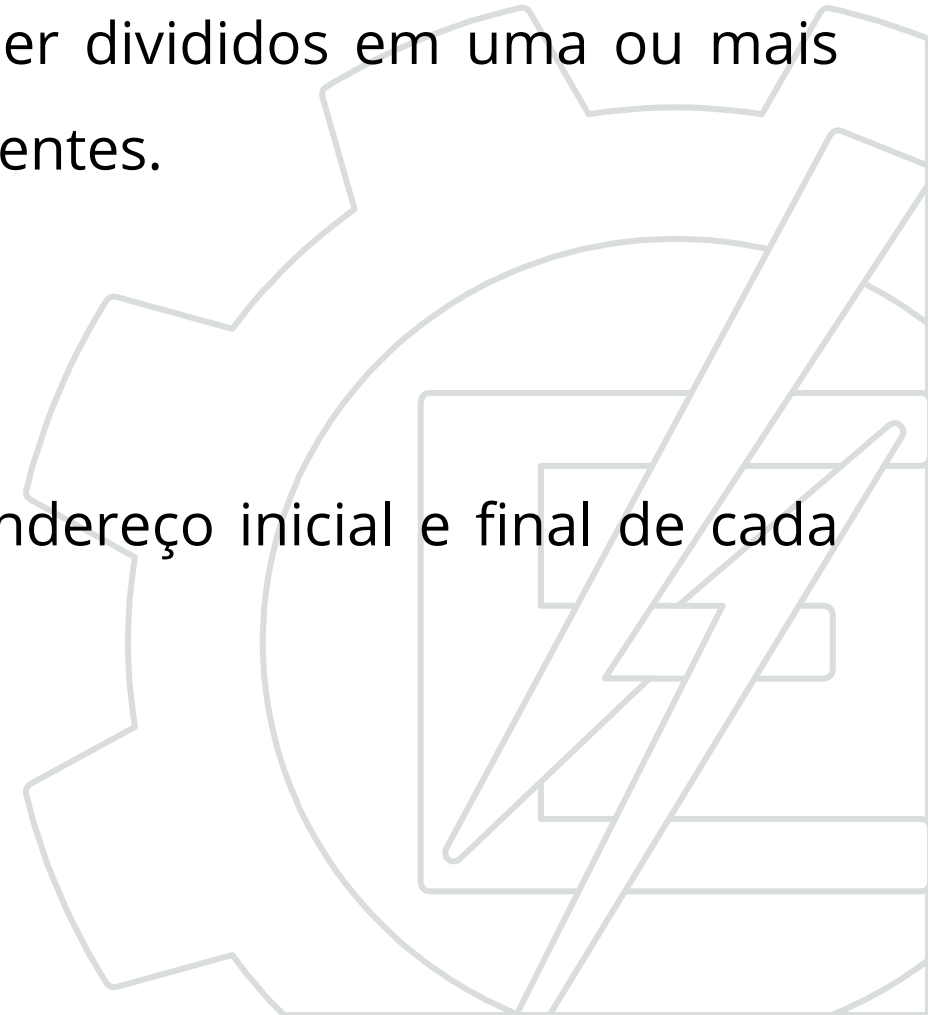
- O sistema de arquivos de disco *Linux Extented File System* conhecida como ext2 e desenvolvida em 1993 pode gerenciar tamanho máximo de 4 TB.
- Em 2001 foi desenvolvido o ext3 com *Journaling*. Em 2006 surge o ext4 aumentando o tamanho máximo do disco para 1 EB. Os ext2, 3 e 4 separam explicitamente as propriedades do arquivo (***inode***) do dado.
- O ext3 possui as seguintes características 1) definição padrão dos atributos de um arquivo; 2) definição padrão do grupo de usuário a um diretório; 3) definição do tamanho da unidade básica de informação na criação do arquivo; 4) checagem da integridade do sistema de arquivos em momentos definidos.

Estrutura de armazenamento - Sistemas de arquivos

- Da mesma forma que um arquivo deve ser aberto antes de ser usado, um sistema de arquivos deve ser montado antes que possa ficar disponível para processos no sistema.
- A estrutura de diretórios pode ser construída em vários volumes que devem ser montados para torná-los disponíveis dentro do espaço de nomes do sistema de arquivos.
- O procedimento de montagem é simples: O sistema operacional recebe o nome do dispositivo e o ponto de montagem — a localização dentro da estrutura de arquivos onde o sistema de arquivos deve ser anexado.

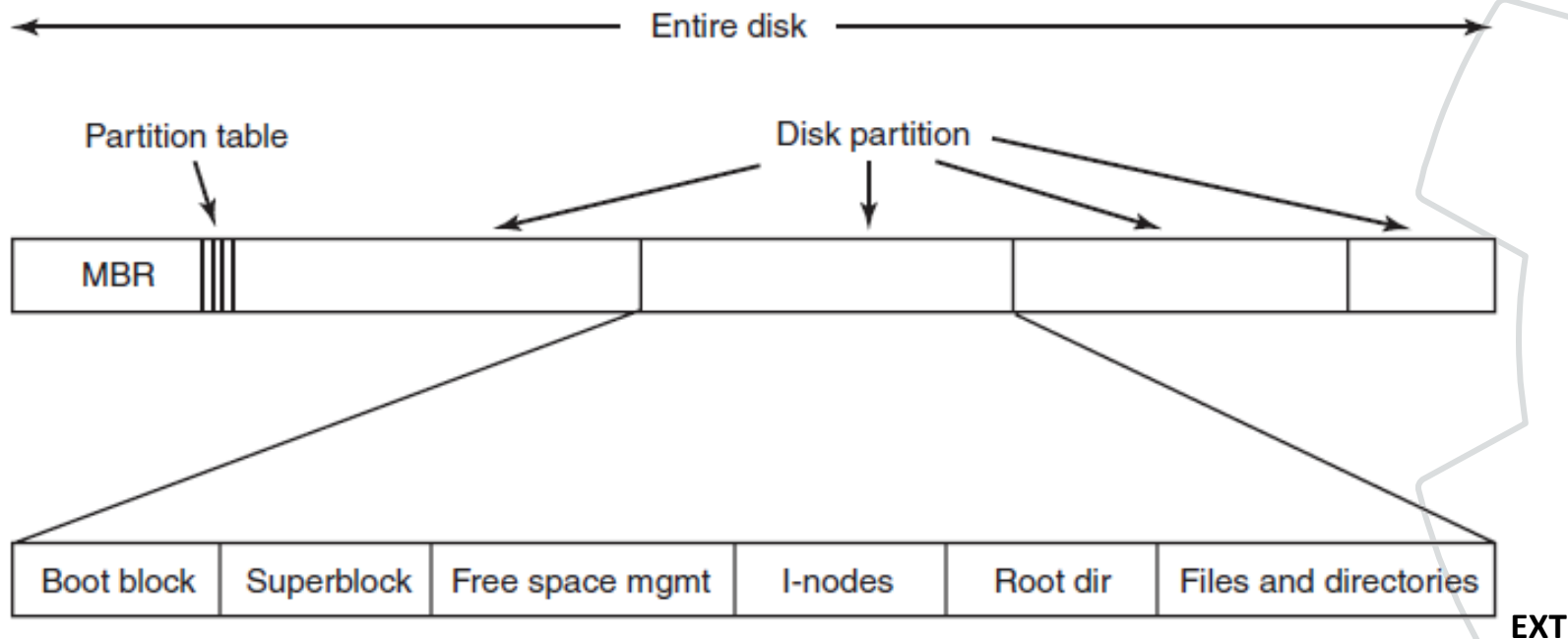
***Layout* do sistema de arquivos**

- Arquivos são armazenados em discos que podem ser divididos em uma ou mais partições e que podem ter sistemas de arquivos diferentes.
- Setor 0 do disco é do MBR (*Master Boot Record*):
 - Usado para iniciar o computador;
 - Onde fica alocada a tabela de partição, com o endereço inicial e final de cada partição, sendo uma delas marcada como ativa;
 - Durante o *boot*, a BIOS lê e executa o MBR.



***Layout* do sistema de arquivos - MBR**

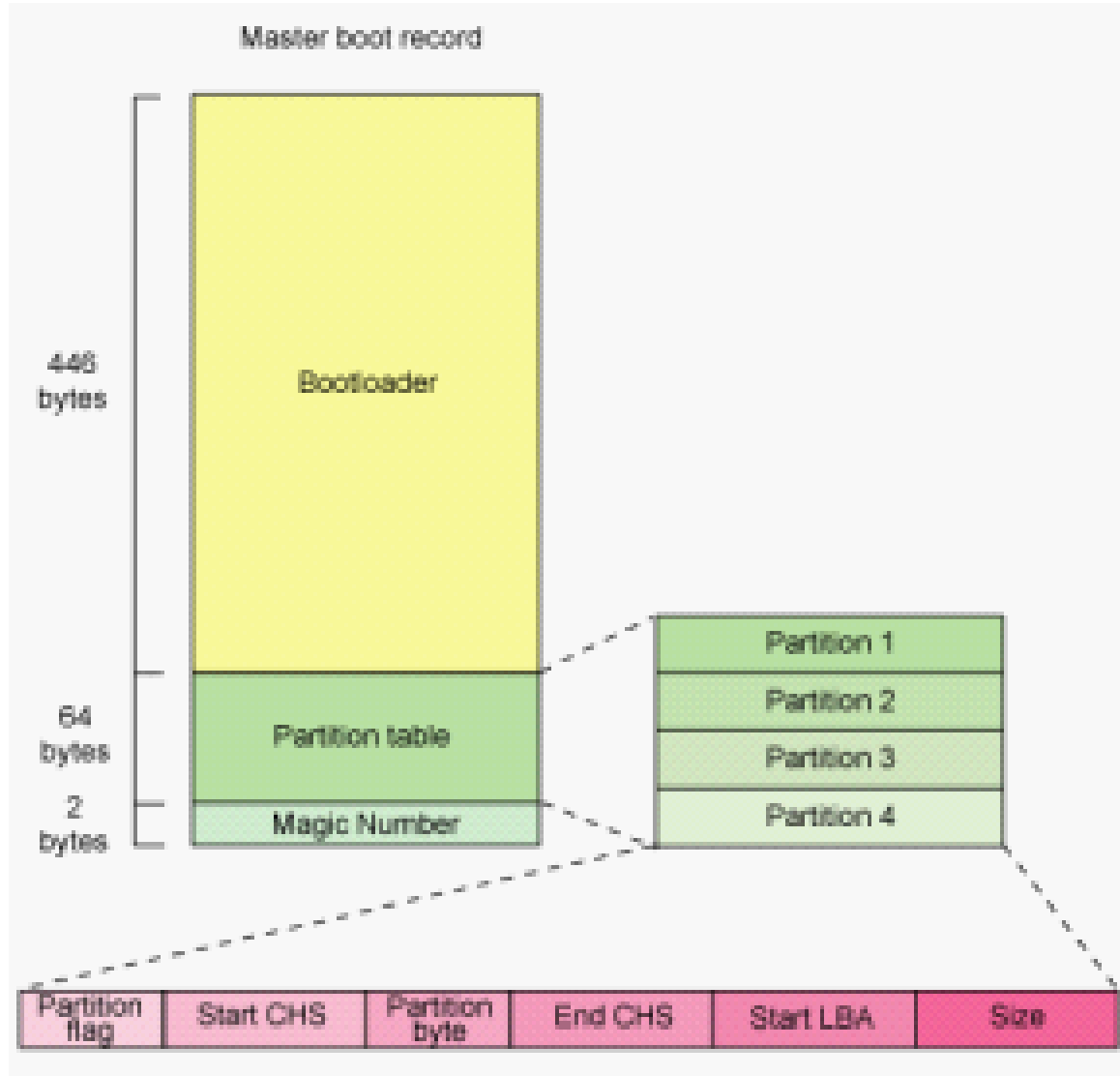
- O programa na MBR localiza a partição ativa, lê seu primeiro bloco (*boot*) e executa-o.
- Toda partição possui um bloco de *boot*.
- O programa no bloco de *boot* carrega o SO daquela partição.



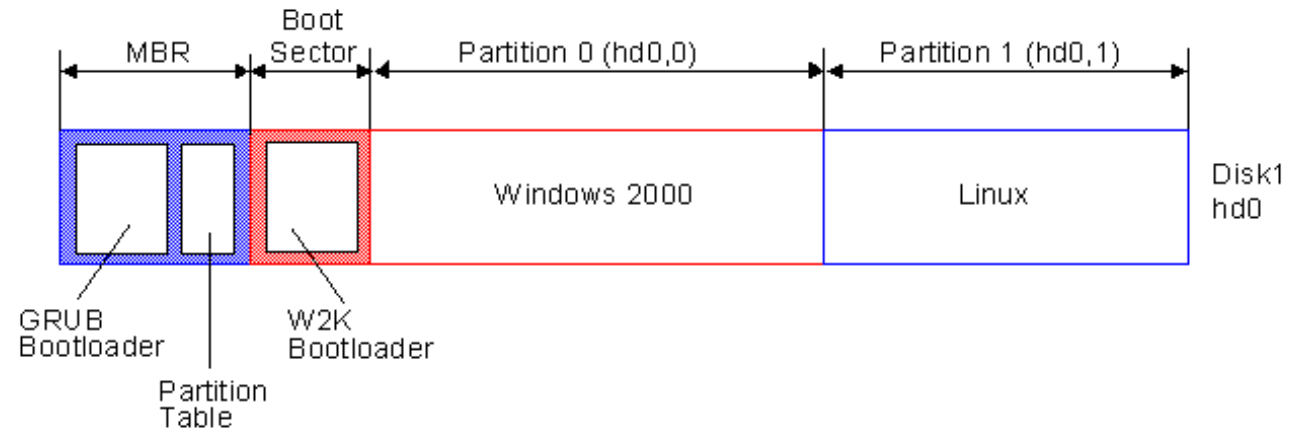
Sistema de Arquivos

Implementação

Layout do sistema de arquivos - MBR

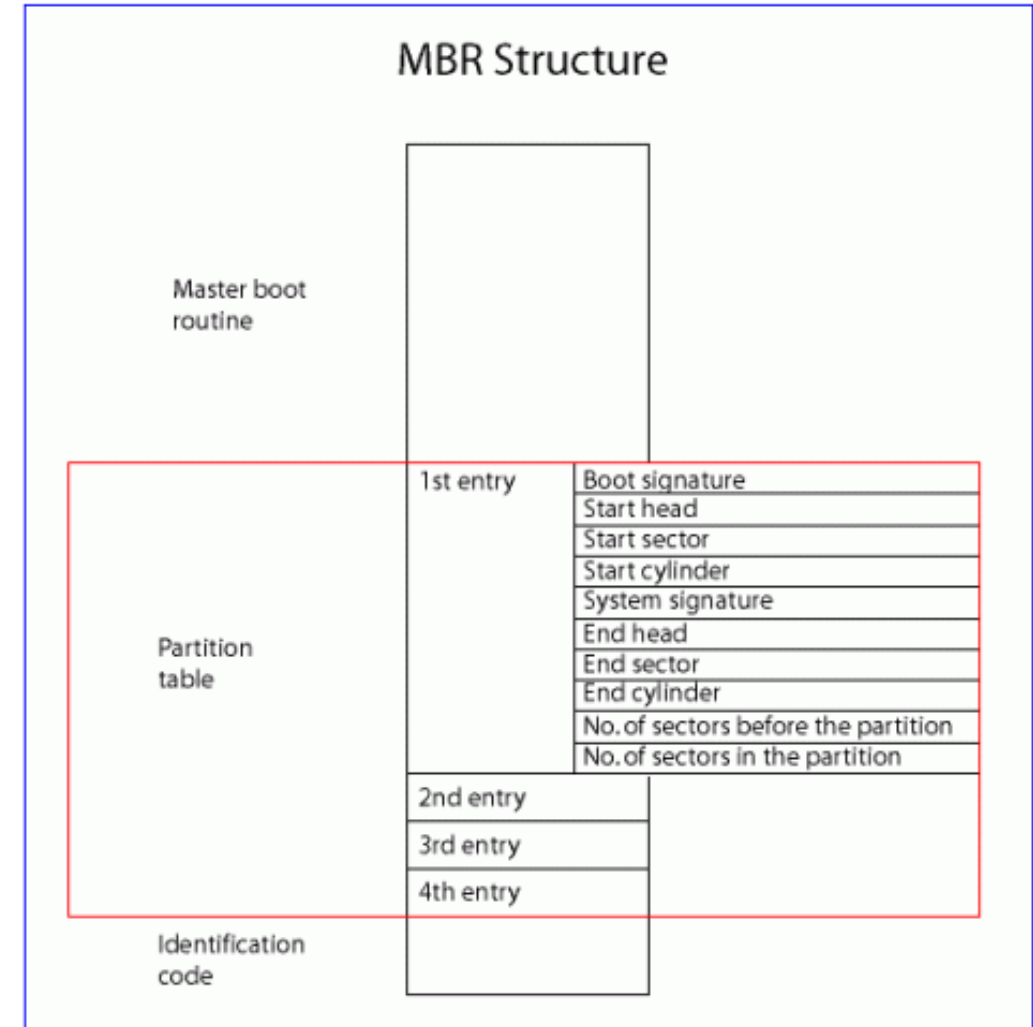
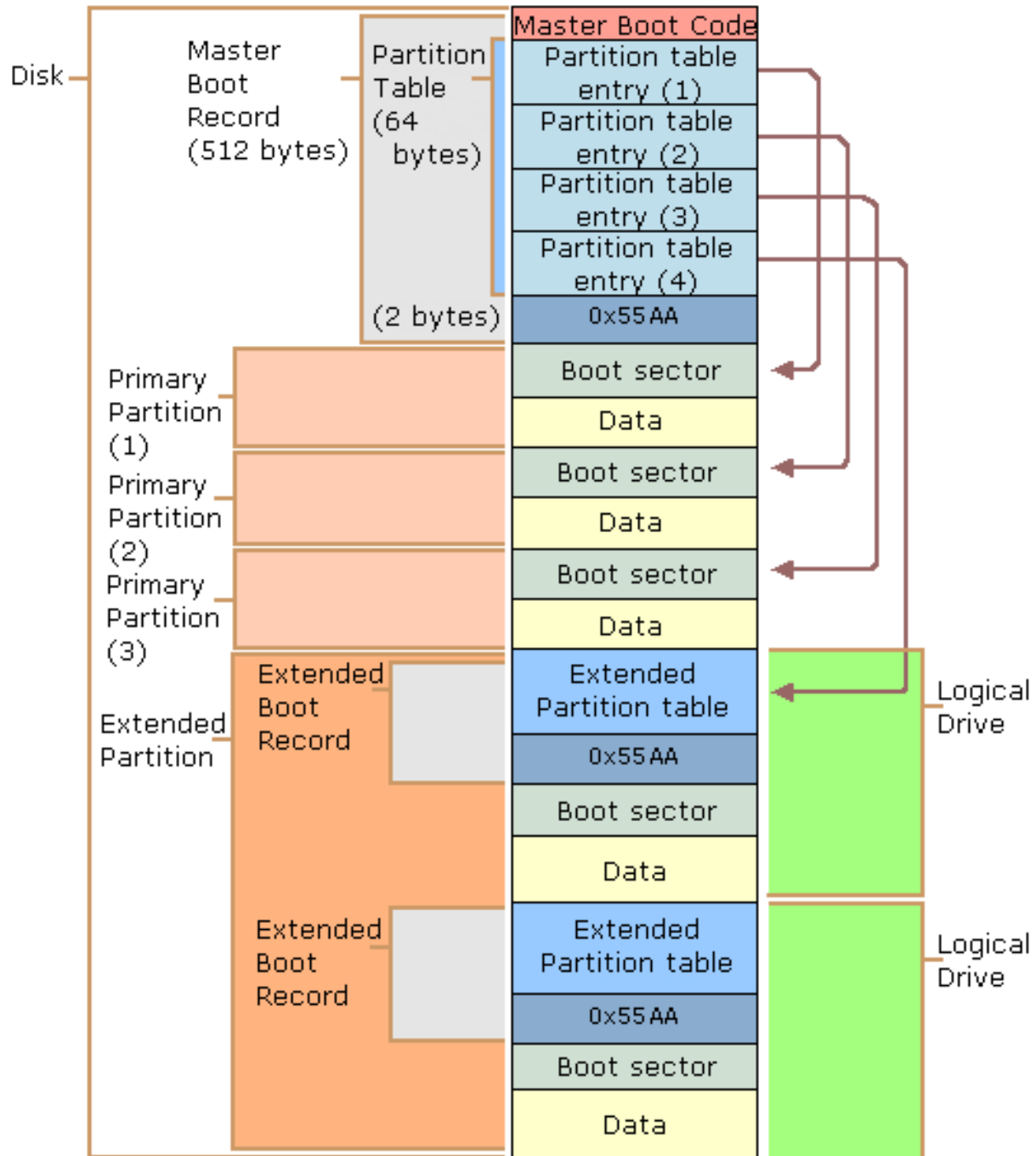


Disk Layout



Sistema de Arquivos

Implementação





MASTER BOOT RECORD

≥ INVOKE-IR

BY: JARED ATKINSON
TEMPLATE BY: ANGE ALBERTINI



BOOT CODE

```
000: 33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00
010: 06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00
020: BD BE 07 80 7E 00 00 7C 0B 0F 85 0E 01 83 C5 10
030: E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00
040: B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09
050: F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74
060: 26 66 68 00 00 00 00 66 FF 76 08 68 00 00 68 00
070: 7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13
080: 9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00
090: 8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1C FE
0A0: 4E 11 75 0C 80 7E 00 80 0F 84 8A 00 B2 80 EB 84
0B0: 55 32 E4 8A 56 00 CD 13 5D EB 9E 81 3E FE 7D 55
0C0: AA 75 6E FF 76 00 E8 8D 00 75 17 FA B0 D1 E6 64
0D0: E8 83 00 B0 DF E6 60 E8 7C 00 B0 FF E6 64 E8 75
0E0: 00 FB B8 00 BB CD 1A 66 23 C0 75 3B 66 81 FB 54
0F0: 43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 BB 00
100: 00 66 68 00 02 00 00 66 68 08 00 00 66 53 66
110: 53 66 55 66 68 00 00 00 66 68 00 7C 00 00 66
120: 61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 00 CD
130: 18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 32 E4
140: 05 00 07 8B F0 AC 3C 00 74 09 BB 07 00 B4 0E CD
150: 10 EB F2 F4 EB FD 2B C9 E4 64 EB 00 24 02 E0 F8
160: 24 02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69
170: 74 69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72
180: 20 6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69
190: 6E 67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E
1A0: 67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74
1B0: 65 6D 00 00 00 63 7B 9A 82 D4 BA 7D 00 00 00 20
1C0: 21 00 07 FE FF FF 00 08 00 00 00 90 36 06 80 FE
1D0: FF FF 07 FE FF FF 00 A0 36 06 00 60 09 00 00 00
1E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA
```

CHS ADDRESSING

```
00100000 00100001 00000000
00100000 100001 0000000000
Head - 1st byte
Sector - 2nd byte (0-5 bits)
Cylinder - 2nd byte (6-7 bits)
3rd byte
```

PARTITION TABLE

PARTITION TYPES

0x00 - EMPTY	0x83 - LINUX
0x01 - FAT12	0x84 - HIBERNATION
0x04 - FAT16	0x85 - LINUX_EXTENDED
0x05 - MS_EXTENDED	0x86 - NTFS_VOLUME_SET
0x06 - FAT16	0x87 - NTFS_VOLUME_SET_1
0x07 - NTFS	0xa0 - HIBERNATION_1
0x0b - FAT32	0xa1 - HIBERNATION_2
0x0c - FAT32	0xa5 - FREEBSD
0x0e - FAT16	0xa6 - OPENBSD
0x0f - MS_EXTENDED	0xa8 - MACOSX
0x11 - HIDDEN_FAT12	0xa9 - NETBSD
0x14 - HIDDEN_FAT16	0xab - MAC_OSX_BOOT
0x16 - HIDDEN_FAT16	0xb7 - BSDI
0x1b - HIDDEN_FAT32	0xb8 - BSDI_SWAP
0x1c - HIDDEN_FAT32	0xee - EFI_GPT_DISK
0x1e - HIDDEN_FAT16	0xef - EFI_SYSTEM_PARTITION
0x42 - MS_MBR_DYNAMIC	0xfb - VMWARE_FILE_SYSTEM
0x82 - SOLARIS_X86	0xfc - VMWARE_SWAP
0x82 - LINUX_SWAP	

END OF MBR

FIELDS

VALUES

jump to boot program
disk parameters
boot program code
disk signature

82D4BA7D

status
starting head
starting sector
starting cylinder
partition type
ending head
ending sector
ending cylinder
relative start sector
total sectors

0x00 - Non-Bootable
0x20
0x21
0x00
0x07 - NTFS
0xFE
0x3F
0x3FF
0x800
0x6369000

status
starting head
starting sector
starting cylinder
partition type
ending head
ending sector
ending cylinder
relative start sector
total sectors

0x80 - Bootable
0xFE
0x3F
0x3FF
0x07 - NTFS
0xFE
0x3F
0x3FF
0x636A000
0x96000

partition type
0x00 - EMPTY

partition type
0x00 - EMPTY

marker
0x55AA

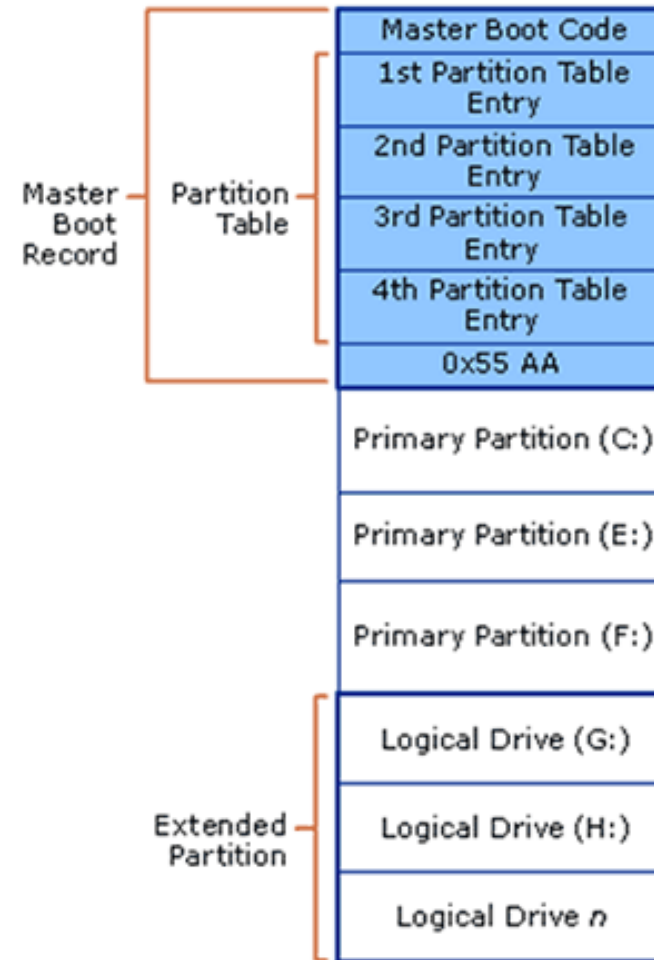
MBR

- Mais antigo;
- Limitado a 2 TB;
- Limitado a 4 partições.

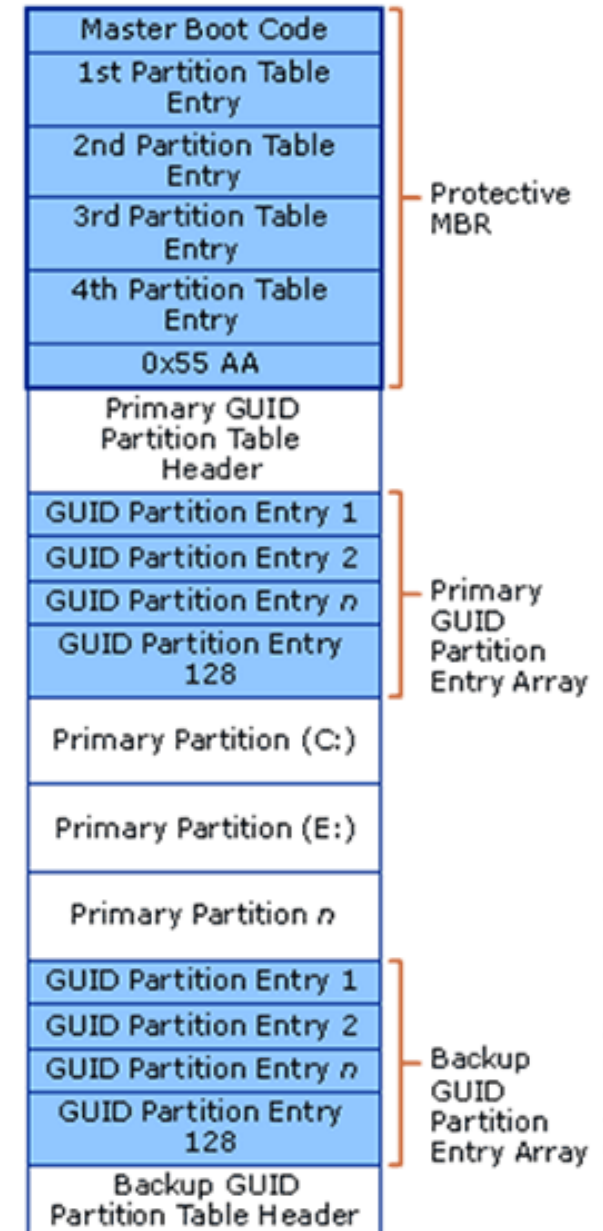
GPT (GUID Partition Table)

- Qualquer tamanho;
- Número ilimitado de partições.
- Windows suporta *boot* utilizando GPT somente utilizando UEFI, em versions de 64-bits

MBR Partition Table Scheme



GPT Partition Table Scheme





GUID PARTITION TABLE



BY: JARED ATKINSON
TEMPLATE BY: ANGE ALBERTINI

PROTECTIVE MBR

FIRST SECTOR OF DRIVE
FOR BREAKDOWN SEE MBR POSTER

```
000 33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00
010 06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00
020 BD BE 07 80 7E 00 00 7C 0B 0F 85 0E 01 83 C5 10
030 E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00
040 B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09
050 F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74
060 26 66 68 00 00 00 66 FF 76 08 68 00 00 68 00
070 7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13
080 9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00
090 8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1C FE
0A0 4E 11 75 0C 80 7E 00 80 0F 84 8A 00 B2 80 EB 84
0B0 55 32 E4 8A 56 00 CD 13 5D EB 9E 81 3E FE 7D 55
0C0 AA 75 6E FF 76 00 E8 8D 00 75 17 FA B0 D1 E6 64
0D0 E8 83 00 80 DF E6 60 E8 7C 00 80 FF E6 64 E8 75
0E0 00 FB 88 00 BB CD 1A 66 23 C0 75 3B 66 81 FB 54
0F0 43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 BB 00
100 00 66 68 00 02 00 00 66 68 08 00 00 00 66 53 66
110 53 66 55 66 68 00 00 00 66 68 00 00 7C 00 00 66
120 61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 00 CD
130 18 A0 B7 07 EB 08 A0 86 07 EB 03 A0 B5 07 32 E4
140 05 00 07 8B F0 AC 3C 00 74 09 BB 07 00 B4 0E CD
150 10 EB F2 F4 EB FD 2B C9 E4 64 EB 00 24 02 E0 F8
160 24 02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69
170 74 69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72
180 20 6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69
190 6E 67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E
1A0 67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74
1B0 65 6D 00 00 00 63 7B 9A 00 00 00 00 00 00 00
1C0 02 00 EE FF FF FF 01 00 00 00 FF FF FF FF 00 00
1D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1F0 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA
```

IMPORTANT PROTECTIVE MBR VALUES

system id EE - EFI GPT partition
GPT header sector offset 1

GPT HEADER

```
200 45 46 49 20 50 41 52 54 00 00 01 00 5C 00 00 00
210 F3 73 9F 97 01 00 00 00 00 00 00 00 00 00 00
220 FF FF 3F 01 00 00 00 00 22 00 00 00 00 00 00 00
230 DE FF 3F 01 00 00 00 00 10 E1 13 F9 35 08 F1 4C
240 96 C7 38 0B 5D B4 A4 2D 02 00 00 00 00 00 00 00
250 80 00 00 00 80 00 00 00 3B 04 A4 F8
```

signature
revision
header size
header CRC32
my LBA
alternate LBA
first usable LBA
last usable LBA
disk guid
partition entry LBA
of partition entries
size of partition entry
partition entry array CRC32

EFI PART
1.0
92
979F73F3
1
20971519
34
20971486
f913e110-0835-4cf1-96c7-380b5db4a42d
2 (sector containing of partition table)
128
128
F8A4043B

PARITION ARRAY

```
400 16 E3 C9 E3 5C 0B 88 4D 81 7D F9 2D F0 02 15 AE
410 47 8A 1A FF F8 08 AB 43 84 10 53 69 7F 0B 23 23
420 22 00 00 00 00 00 00 00 21 00 01 00 00 00 00 00
430 00 00 00 00 00 00 00 00 4D 00 69 00 63 00 72 00
440 6F 00 73 00 6F 00 66 00 74 00 20 00 72 00 65 00
450 73 00 65 00 72 00 76 00 65 00 64 00 20 00 70 00
460 61 00 72 00 74 00 69 00 74 00 69 00 6F 00 6E 00
470 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

partition type guid
unique partition guid
starting LBA
ending LBA
attributes
partition name

e3c9e316-0b5c-4db8-817d-f92df00215ae
ff1a8a47-08f8-43ab-b410-53697f0b2323
34
65569
0
Microsoft reserved partition

```
480 A2 A0 D0 EB E5 B9 33 44 87 C0 68 B6 B7 26 99 C7
490 42 AE 76 6D C1 B6 BE 4F 8D 42 20 CD 36 60 26 B4
4A0 00 08 01 00 00 00 00 00 FF 07 00 00 00 00 00 00
4B0 00 00 00 00 00 00 00 00 42 00 61 00 73 00 69 00
4C0 63 00 20 00 64 00 61 00 74 00 61 00 20 00 70 00
4D0 61 00 72 00 74 00 69 00 74 00 69 00 6F 00 6E 00
4E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

partition type guid
unique partition guid
starting LBA
ending LBA
attributes
partition name

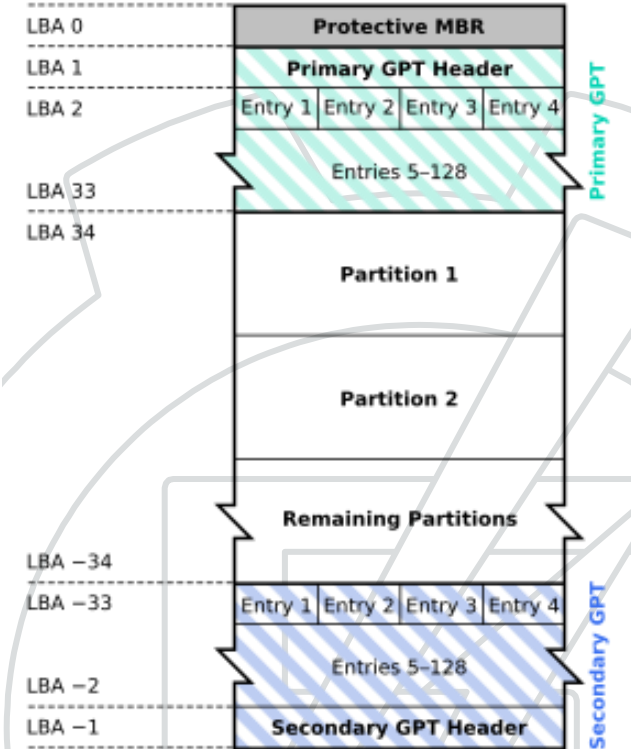
ebd0a0a2-b9e5-4433-87c0-68b6b72699c7
6d76ae42-b6c1-4fbe-8d42-20cd366026b4
67584
2164735
0
Basic data partition

```
500 A2 A0 D0 EB E5 B9 33 44 87 C0 68 B6 B7 26 99 C7
510 3A 5C 79 D6 4D 8A B4 4F 91 A0 48 88 12 CC E0 27
520 00 08 00 00 00 00 00 00 FF 07 41 00 00 00 00 00
530 00 00 00 00 00 00 00 00 42 00 61 00 73 00 69 00
540 63 00 20 00 64 00 61 00 74 00 61 00 20 00 70 00
550 61 00 72 00 74 00 69 00 74 00 69 00 6F 00 6E 00
560 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
570 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

partition type guid
unique partition guid
starting LBA
ending LBA
attributes
partition name

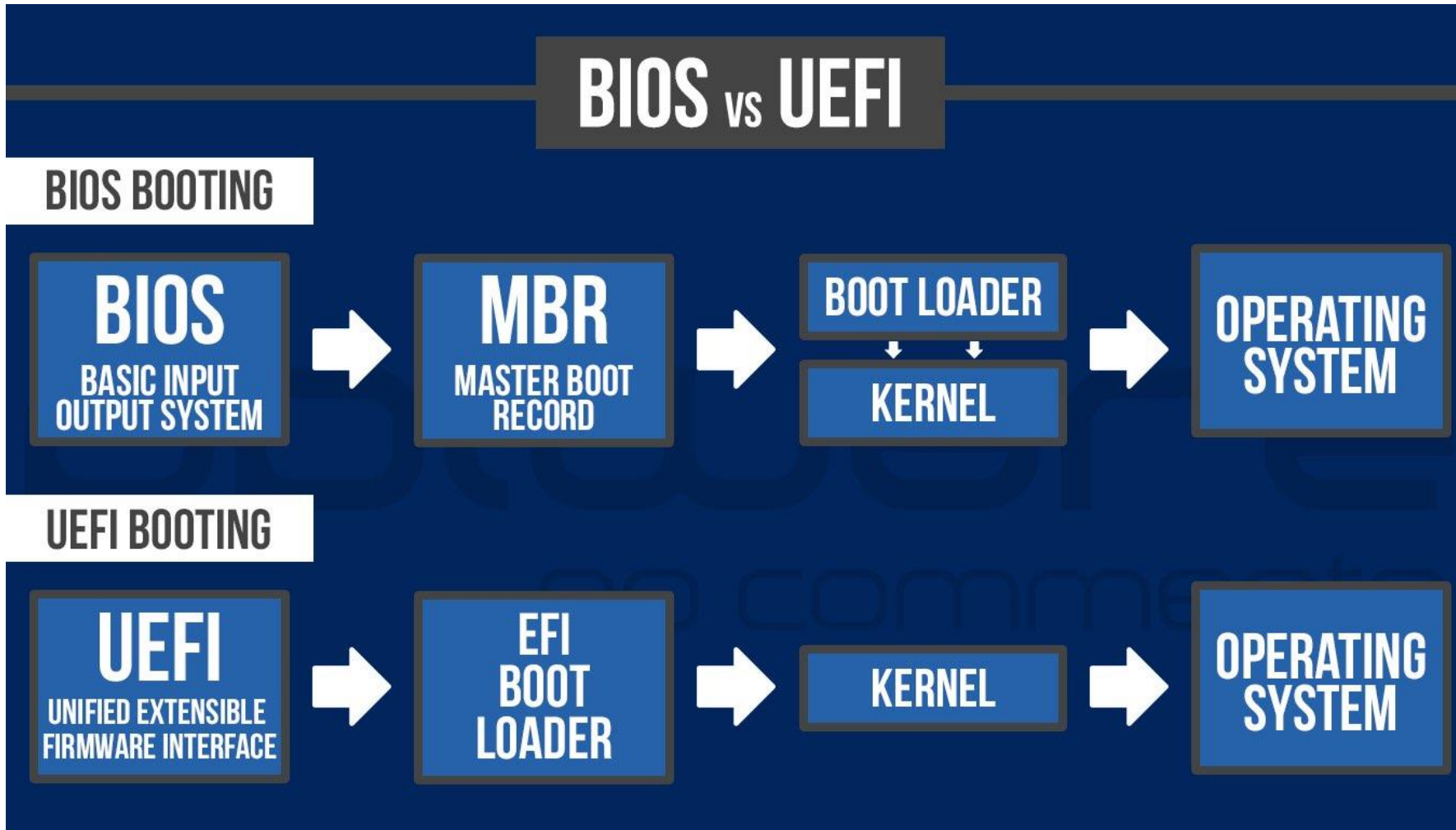
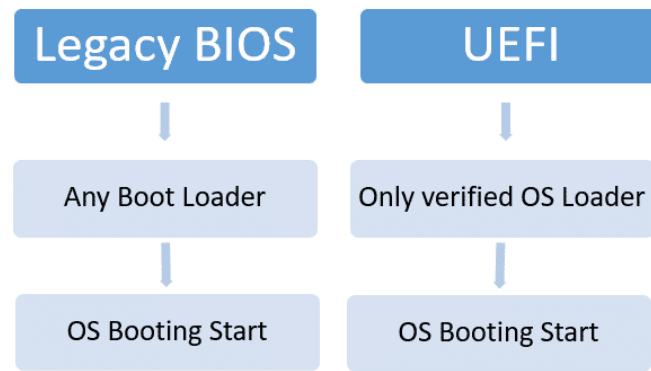
ebd0a0a2-b9e5-4433-87c0-68b6b72699c7
d6795c3a-8a4d-4fb4-91a0-488812cce027
2164736
4261887
0
Basic data partition

GUID Partition Table Scheme



Sistema de Arquivos

Implementação



Sistema de Arquivos

Proteção

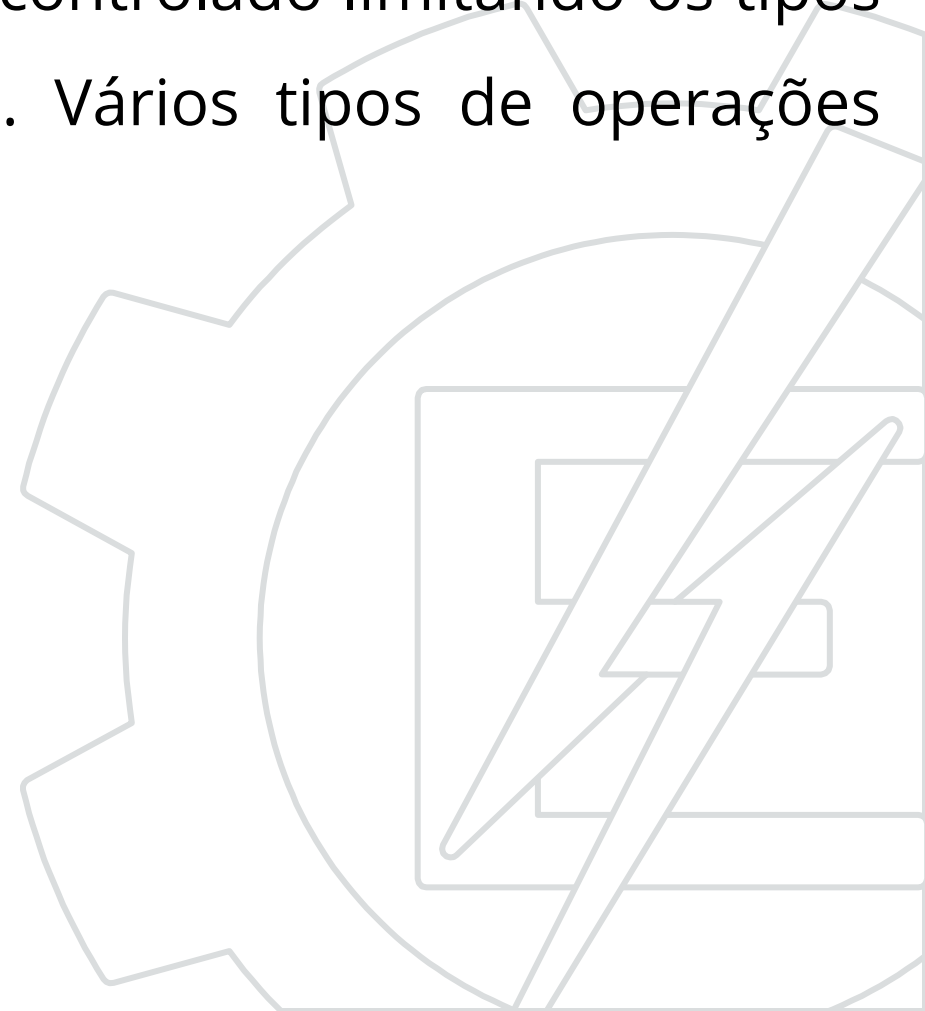


Proteção do sistema de arquivos

- Quando informações são armazenadas em um sistema de computação, queremos mantê-las protegidas contra **danos físicos** (a questão da confiabilidade) e **acesso indevido** (a questão da proteção).
- A **confiabilidade** geralmente é fornecida por cópias duplicadas dos arquivos. Arquivos podem ser excluídos acidentalmente. *Bugs* no software do sistema de arquivos também podem fazer com que o conteúdo dos arquivos seja perdido.
- A **proteção** pode ser fornecida de muitas maneiras. Em um sistema multiusuário pode ser utilizado o controle de acesso e as permissões.

Tipos de acesso ao sistema de arquivos

- Os mecanismos de proteção fornecem acesso controlado limitando os tipos de acesso a arquivos que podem ser feitos. Vários tipos de operações diferentes podem ser controladas:
 - Leitura.
 - Gravação.
 - Execução.
 - Acréscimo.
 - Exclusão.
 - Listagem do nome e dos atributos dos arquivos.



Controle de acesso ao sistema de arquivos

- A abordagem mais comum relacionada com o problema da proteção é tornar o acesso dependente da identidade do usuário. Diferentes usuários podem precisar de diferentes tipos de acesso a um arquivo ou diretório.
- O esquema mais geral para a implementação do acesso dependente da identidade é associar a cada arquivo e diretório uma lista de controle de acesso (**ACL** — *access control list*) especificando nomes de usuários e os tipos de acesso permitidos a cada usuário

Controle de acesso ao sistema de arquivos - ACL

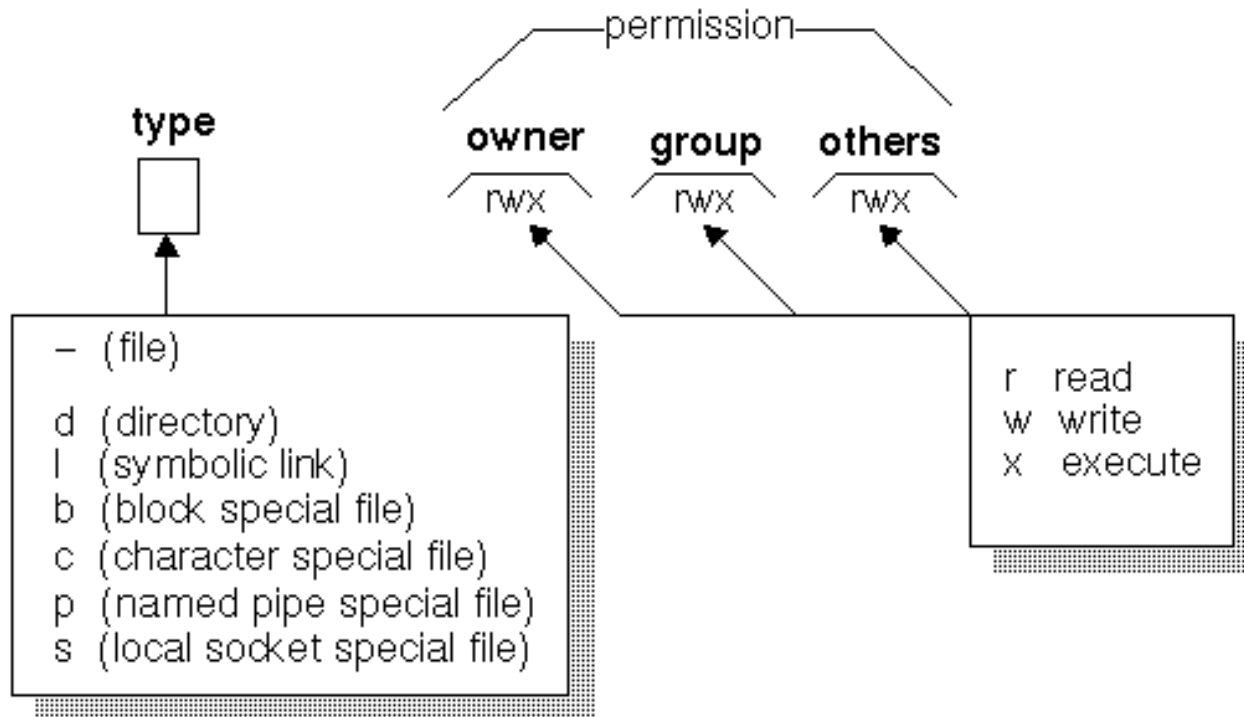
- O principal problema das listas de acesso é seu tamanho. Se quisermos permitir que todos leiam um arquivo, teremos que listar todos os usuários com acesso de leitura.
- Essa técnica tem duas consequências indesejáveis:
 - A construção desse tipo de lista pode ser uma tarefa tediosa e pouco compensadora, principalmente se não conhecermos antecipadamente a lista de usuários no sistema.
 - A entrada do diretório, anteriormente de tamanho fixo, agora deve ser de tamanho variável, resultando em gerenciamento de espaço mais complicado.

Controle de acesso ao sistema de arquivos - *ACL*

- Esses problemas podem ser resolvidos com o uso de uma versão condensada da lista de acesso. Para condensar o tamanho da lista de controle de acesso, muitos sistemas reconhecem três classificações de usuários associadas a cada arquivo:
 - **Proprietário.** O usuário que criou o arquivo é o proprietário.
 - **Grupo.** Um conjunto de usuários que estão compartilhando o arquivo e precisam de acesso semelhante é um grupo, ou grupo de trabalho.
 - **Universo.** Todos os outros usuários no sistema constituem o universo.

Controle de acesso ao sistema de arquivos - ACL

- Esses problemas podem ser resolvidos com o uso de uma versão condensada da lista de acesso. Para condensar o tamanho da lista de controle de acesso, muitos sistemas reconhecem três classificações de usuários associadas a cada arquivo:



Sistema de Arquivos

Layout

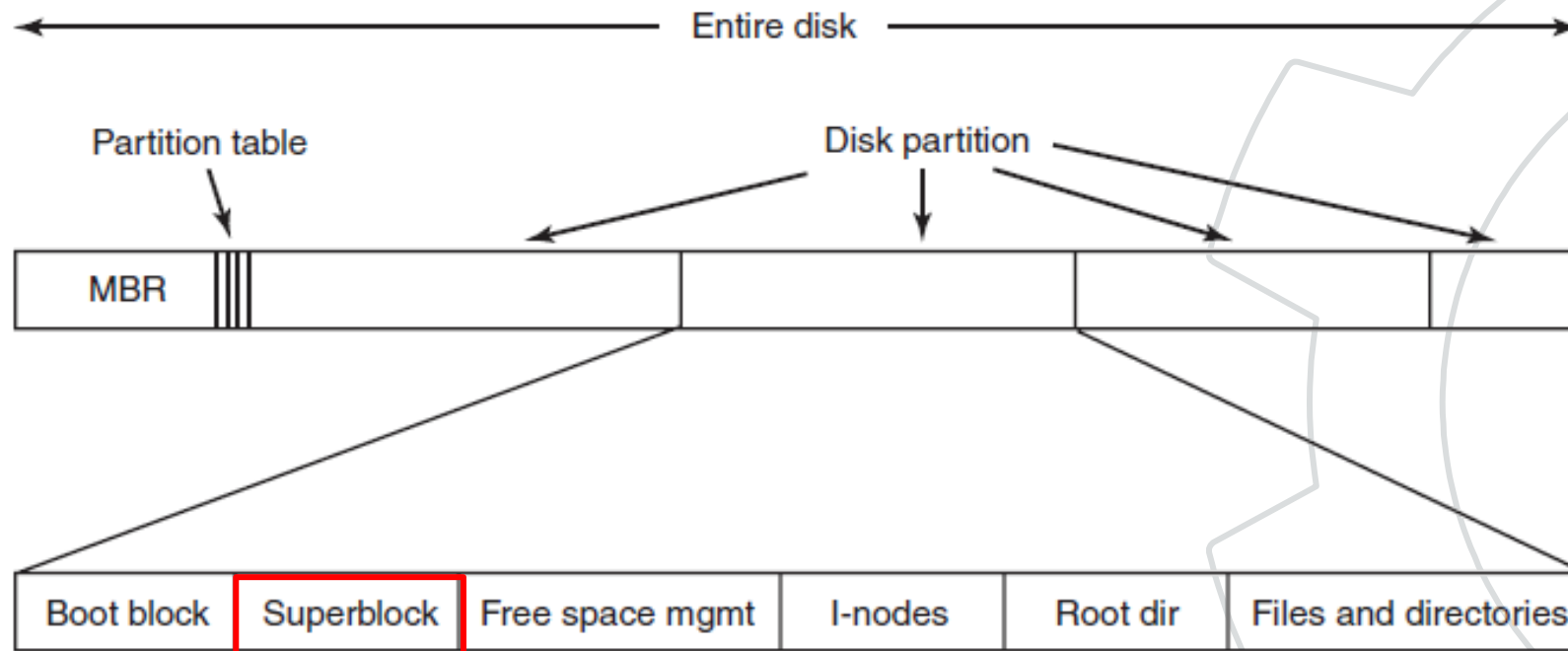


Sistema de Arquivos

Layout

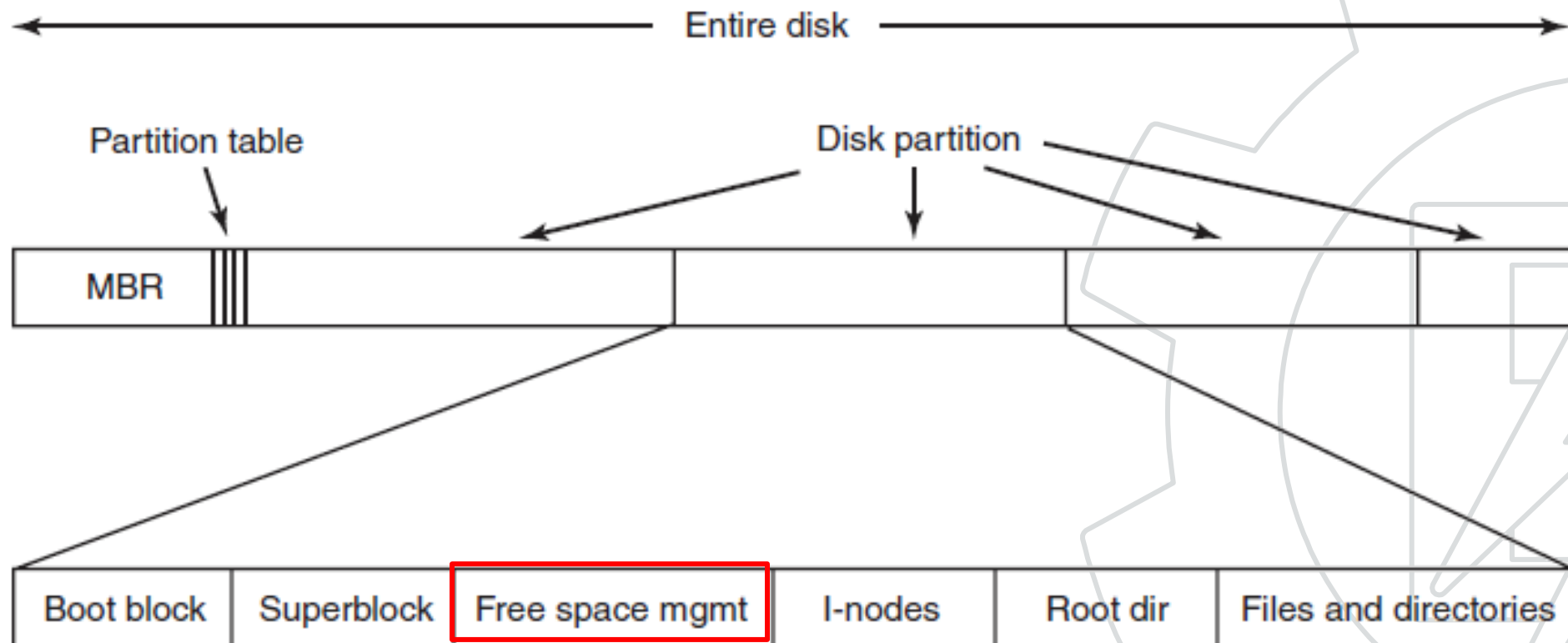
Layout do sistema de arquivos - MBR

- O **superblock** contém os parâmetros-chave, como tamanho e número de blocos, ID do sistema de arquivos (conhecido como número mágico), além de outros campos.
- É lido para a memória no boot ou quando este sistema de arquivos é manipulado pela primeira vez.



Layout do sistema de arquivos - MBR

- Informação a respeito de ***blocos livres*** é armazenado através de mapa de bits ou de lista de ponteiros.

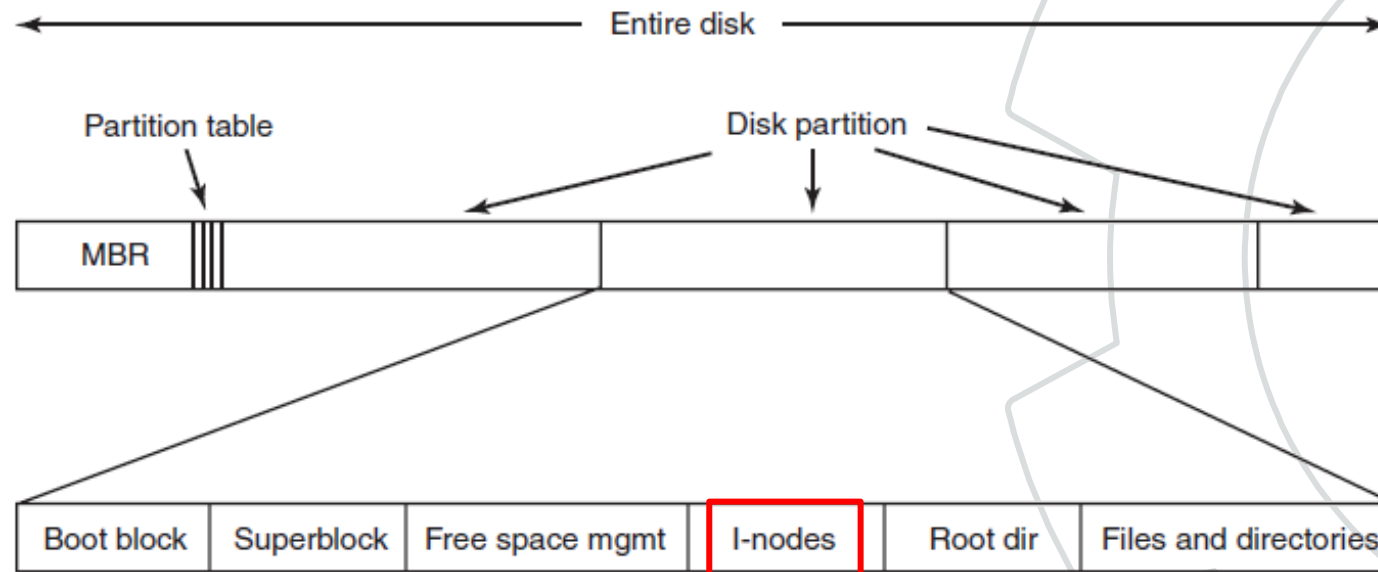


Sistema de Arquivos

Layout

Layout do sistema de arquivos - MBR

- A estrutura de dados dos arquivos é representada pelos ***i-nodes***.
- ***i-node*** - *Index-node*.
- Existe uma por arquivo, com informações e localização.
- Utilizada no Unix/Linux.

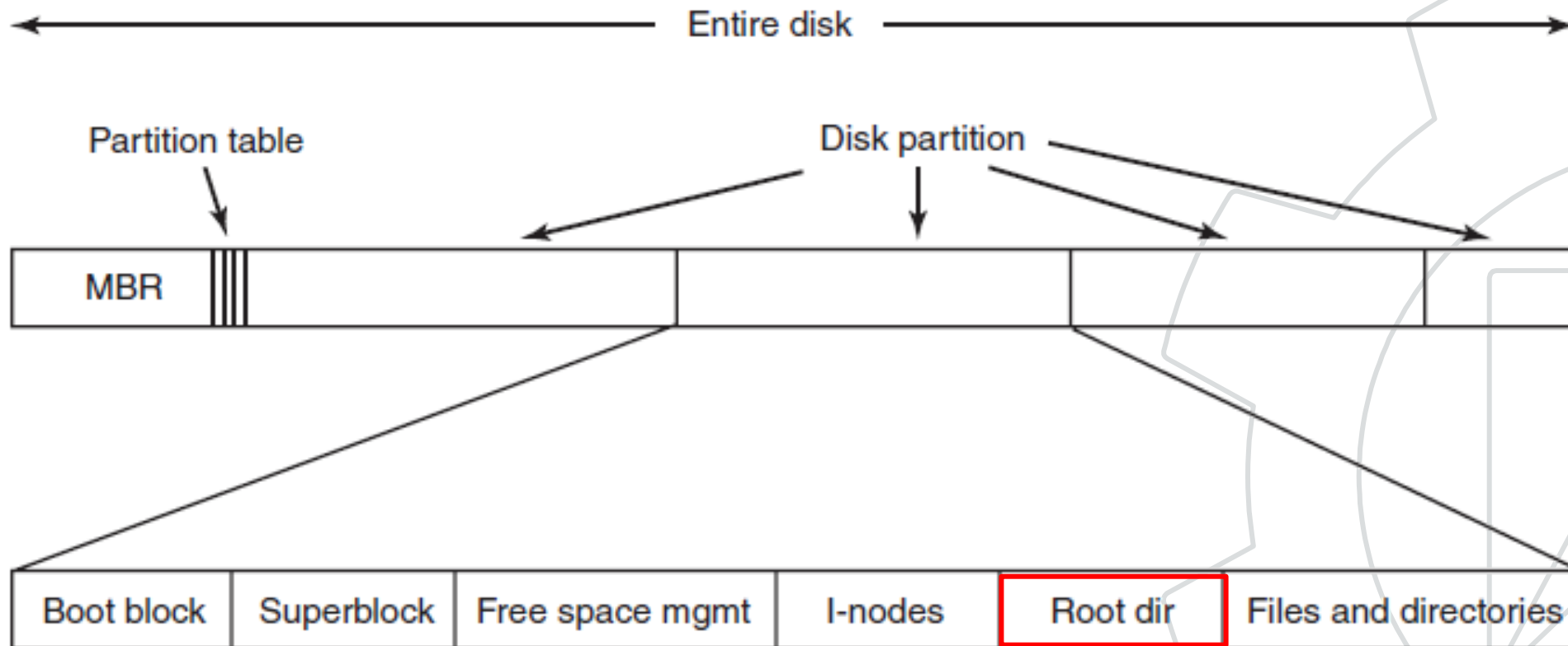


Sistema de Arquivos

Layout

Layout do sistema de arquivos - MBR

- Diretório-raiz (**root**) contém o topo da árvore do sistema de arquivos.

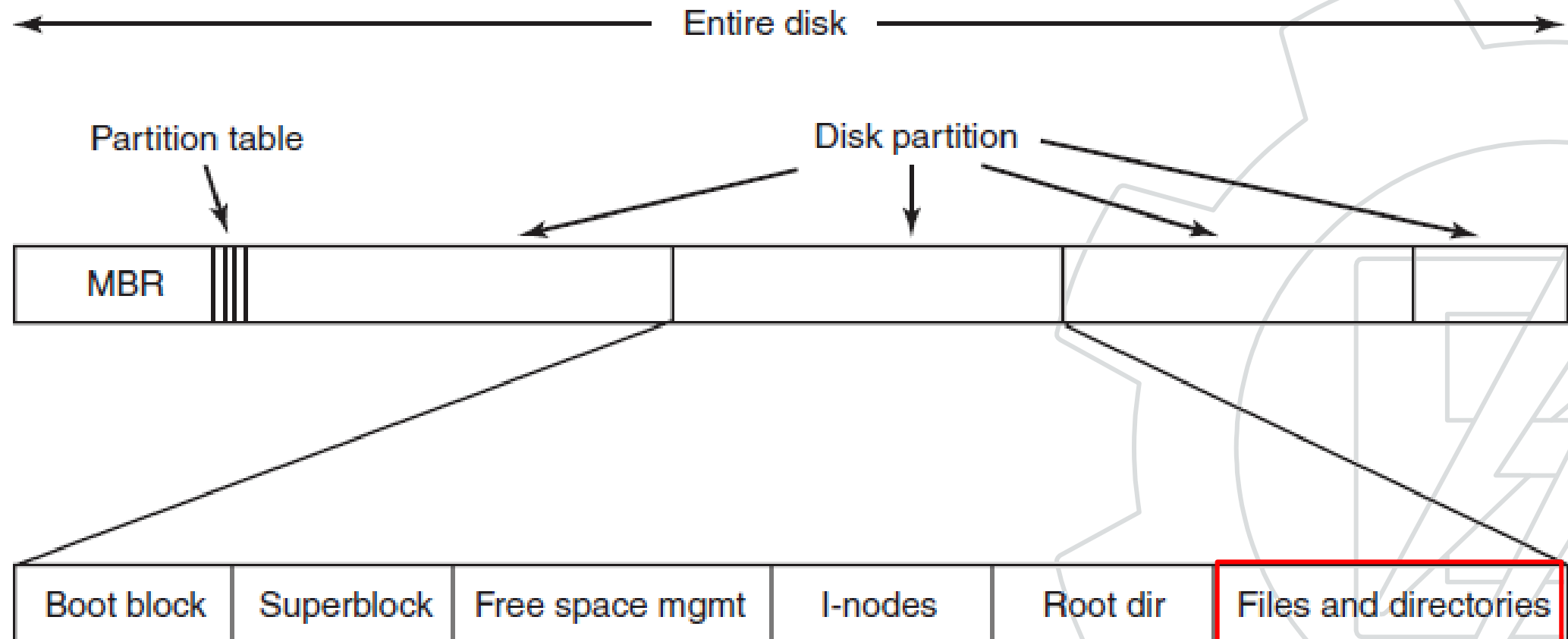


Sistema de Arquivos

Layout

Layout do sistema de arquivos - MBR

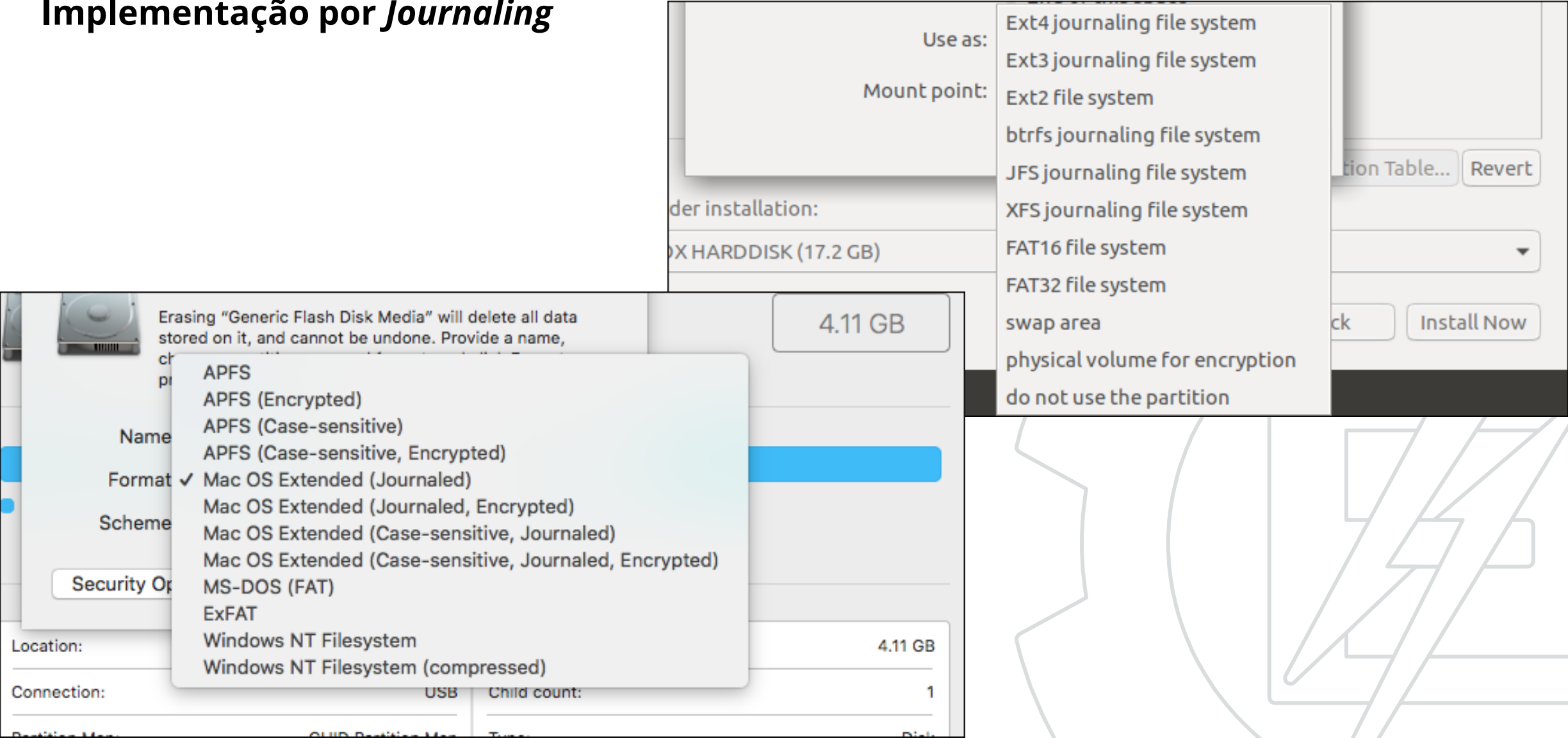
- Demais arquivos e diretórios da partição.



Sistema de Arquivos

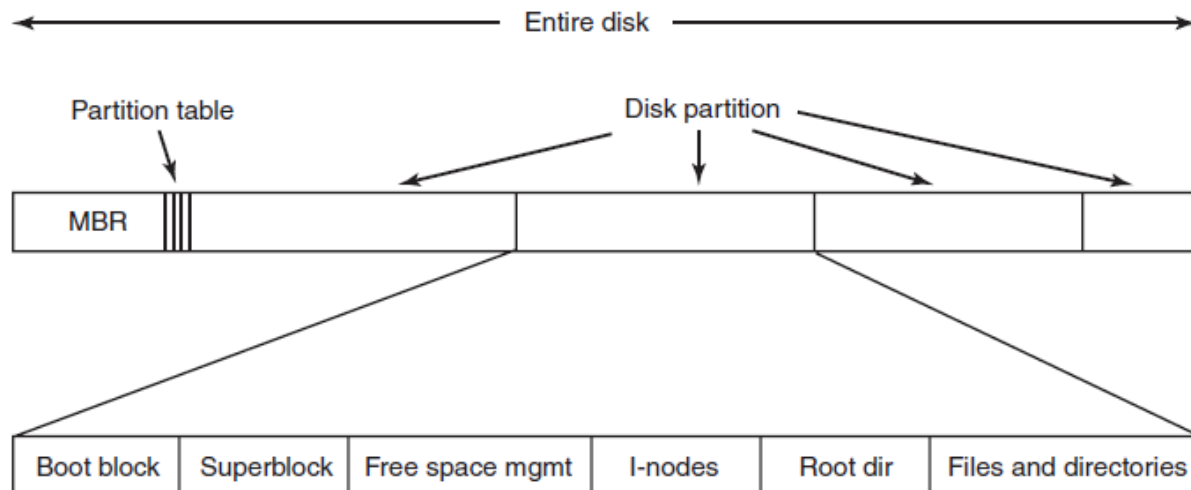
Implementação

Implementação por *Journaling*



Implementação por *Journaling*

- Considere a remoção de um arquivo no Unix:
 - 1) Remoção do arquivo de seu diretório;
 - 2) Liberação do *i-node* para o conjunto de *i-nodes* livres;
 - 3) Devolução dos blocos livres no disco.



Implementação por *Journaling*

- Considere a remoção de um arquivo no Unix:
 - 1) Remoção do arquivo de seu diretório;
 - 2) Liberação do *i-node* para o conjunto de *i-nodes* livres;
 - 3) Devolução dos blocos livres no disco.
- E se o sistema parar ao final da primeira etapa?
 - Existirão *i-nodes* com blocos não acessíveis e com endereços não-relocáveis.
- E ao final da segunda etapa?
 - Os blocos serão perdidos.

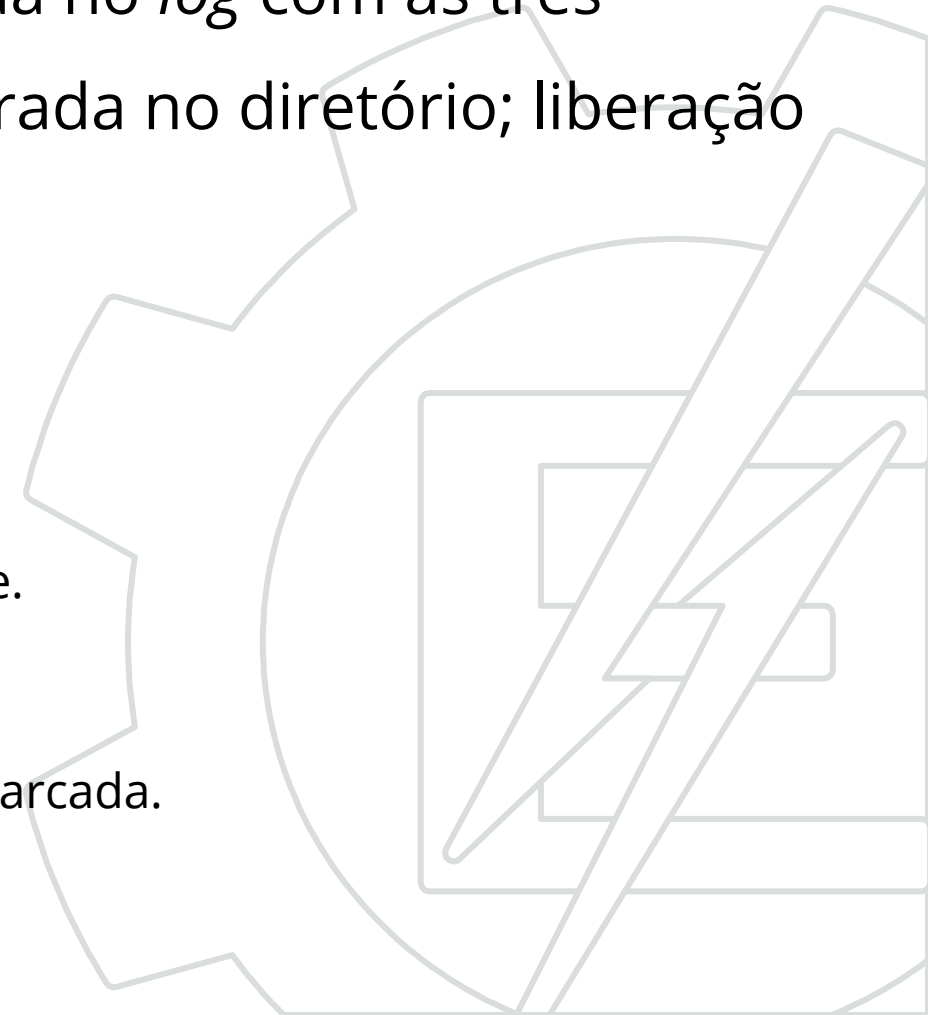


Implementação por *Journaling*

- Técnica para criar uma certa robustez diante das falhas.
- Mantém-se um ***log – journal***.
- Controla as informações do que o sistema de arquivos irá fazer antes que ele efetivamente o faça (*checkpoints*). Caso o sistema falhe antes de terminar a execução do trabalho, conseguirá finalizá-la após a inicialização.
- Sistemas de arquivo *Journaling*:
 - Windows – ***NTFS***
 - Linux – ***ext3*** em diante

Implementação por *Journaling* - Funcionamento

- Conforme o exemplo apresentado: uma entrada no *log* com as três operações a serem realizadas: remoção da entrada no diretório; liberação dos i-nodes e dos blocos.
- Procedimento:
 - Grava o *log* no disco;
 - Lê de volta para a memória, para verificar a sua integridade.
 - Só então as operações têm início.
 - Quando uma operação é concluída, sua entrada no *log* é marcada.
 - *Logs* são excluídos periodicamente ou quando cheios.

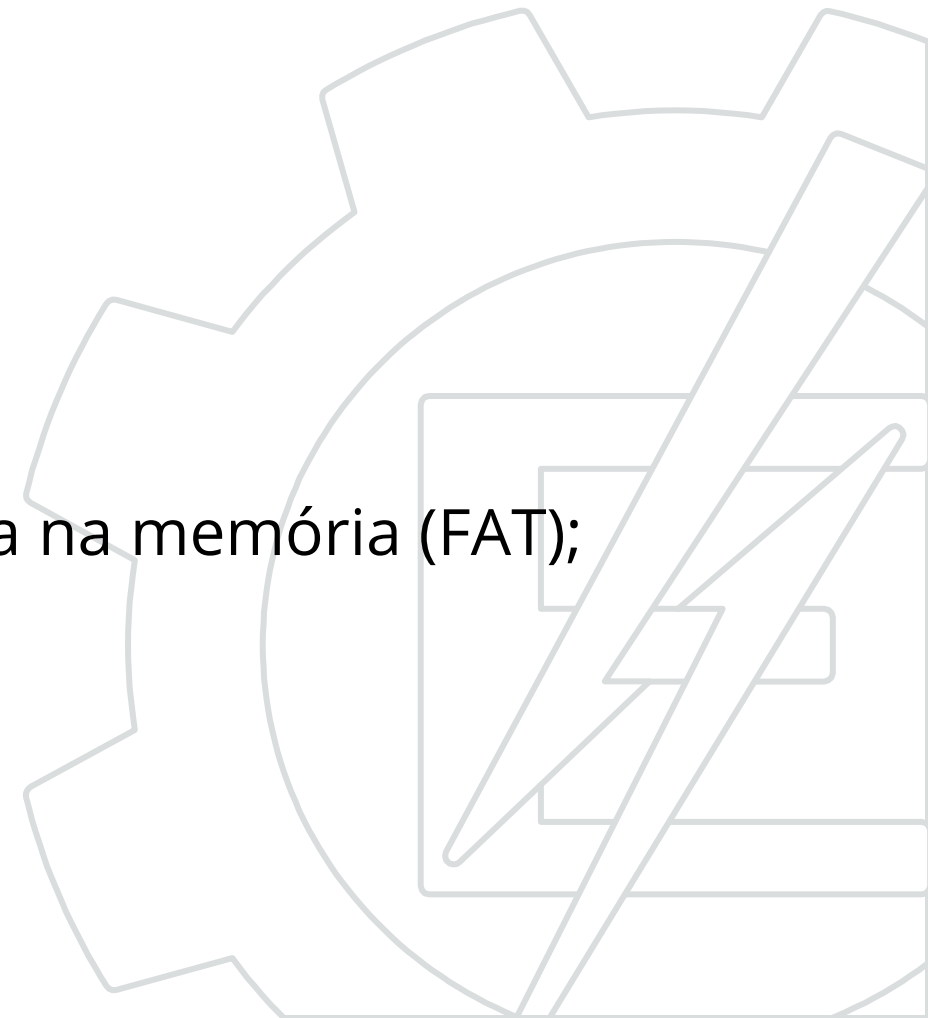


Implementação por *Journaling*

- Sistemas de arquivo *journaling* devem organizar suas estruturas de dados e operações no *log* de forma que todos sejam idempotentes, isto é, suas ações devem ser repetidas sempre que necessário sem causarem nenhum dano.
- Exemplo:
 - Desejo incluir novos blocos ao final da lista de blocos livres e não sei se eles já foram incluídos. Os blocos podem já estar lá.
 - Deve-se, antes, pesquisar se já não estão lá e, em caso negativo, incluí-los.

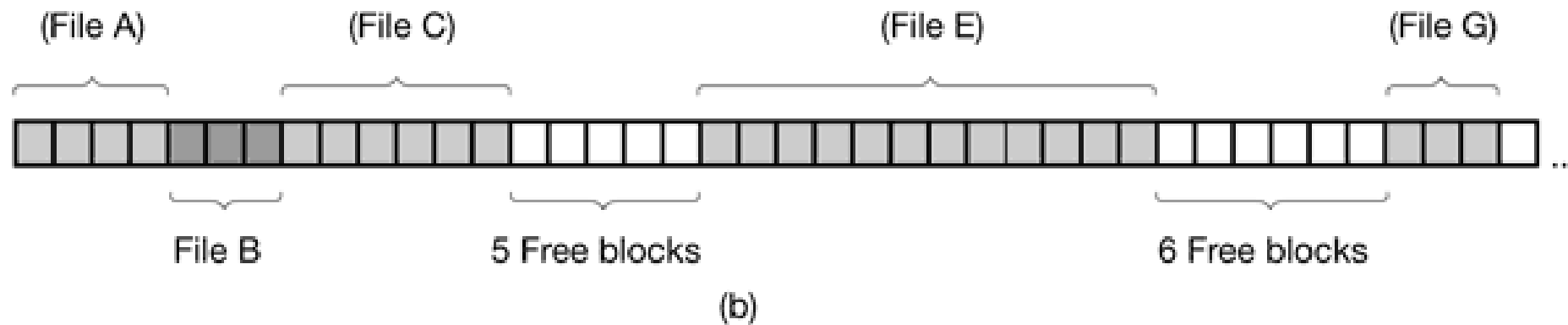
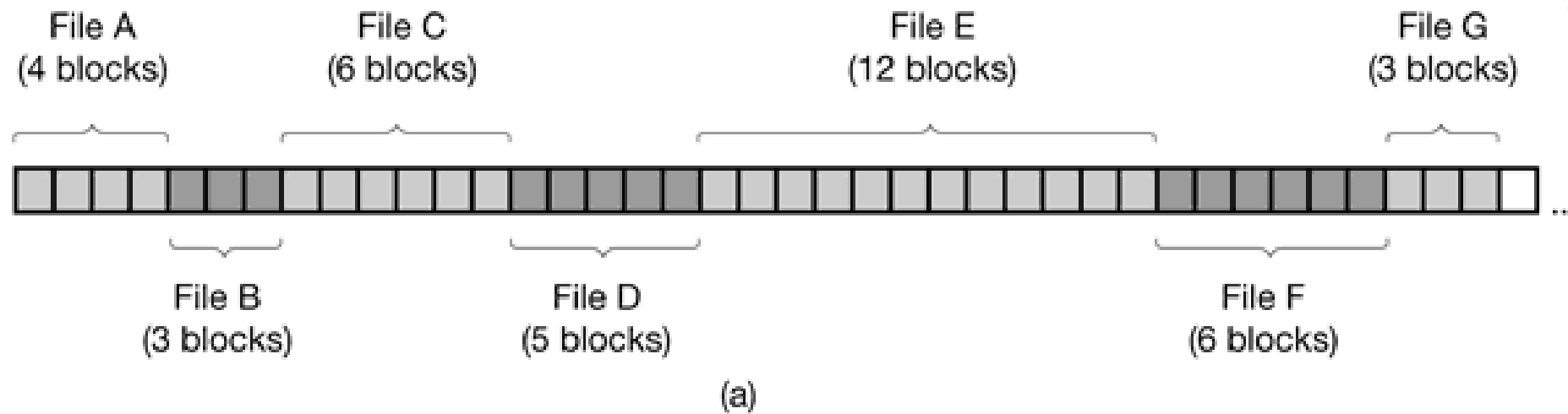
Implementando arquivos

- Como os arquivos são alocados no disco?
- Diferentes técnicas por diferentes SOs:
 - Alocação contínua;
 - Alocação com lista encadeada;
 - Alocação com lista enc. utilizando uma tabela na memória (FAT);
 - *i-nodes*.



Implementando arquivos - Alocação contínua

- Técnica mais simples - Armazena arquivos de forma contínua no disco.

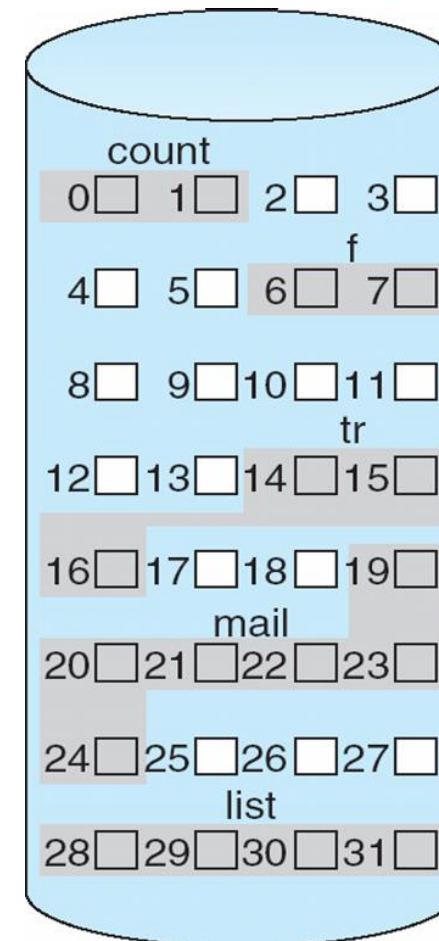


Implementando arquivos

Alocação contínua/contígua

- **Vantagens:**

- Simples de implementar – Necessário saber o endereço do primeiro bloco e o número de blocos.
- Alto desempenho na leitura.
- Utiliza um *seek* para o primeiro bloco e segue a leitura sequencial.



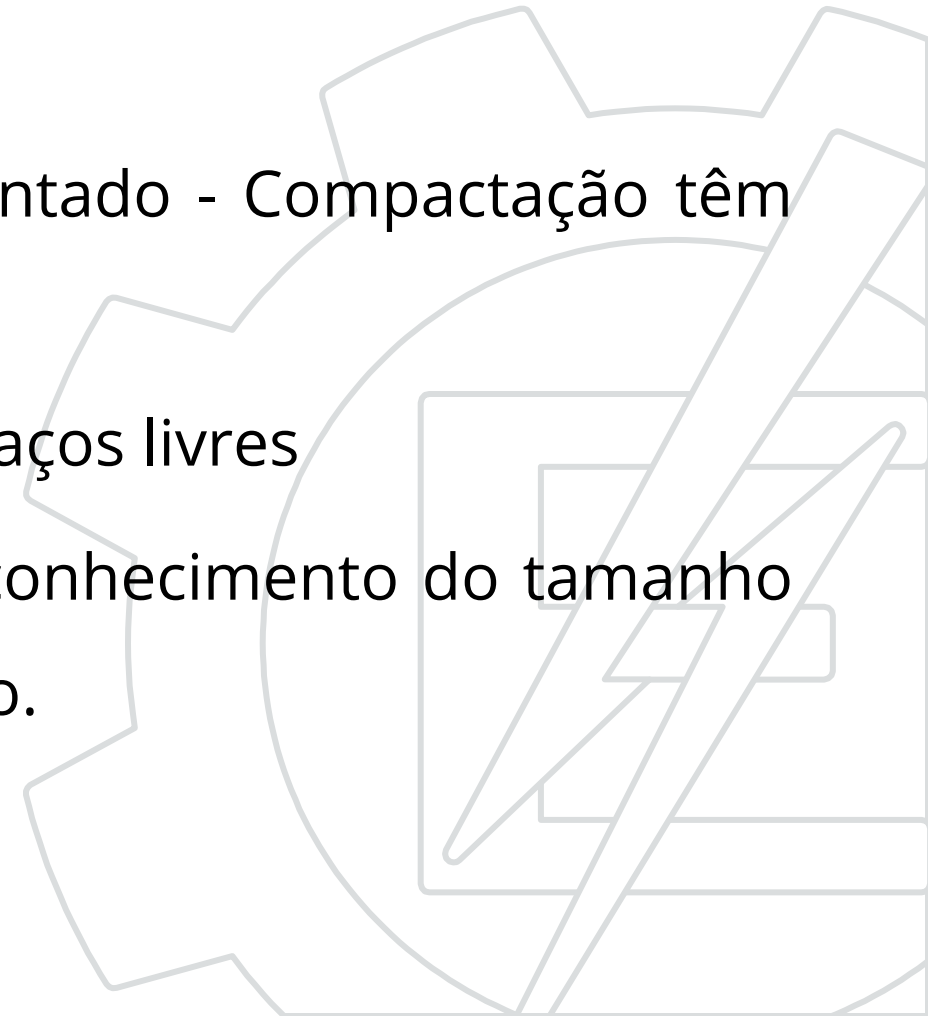
directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Implementando arquivos - Alocação contínua/contígua

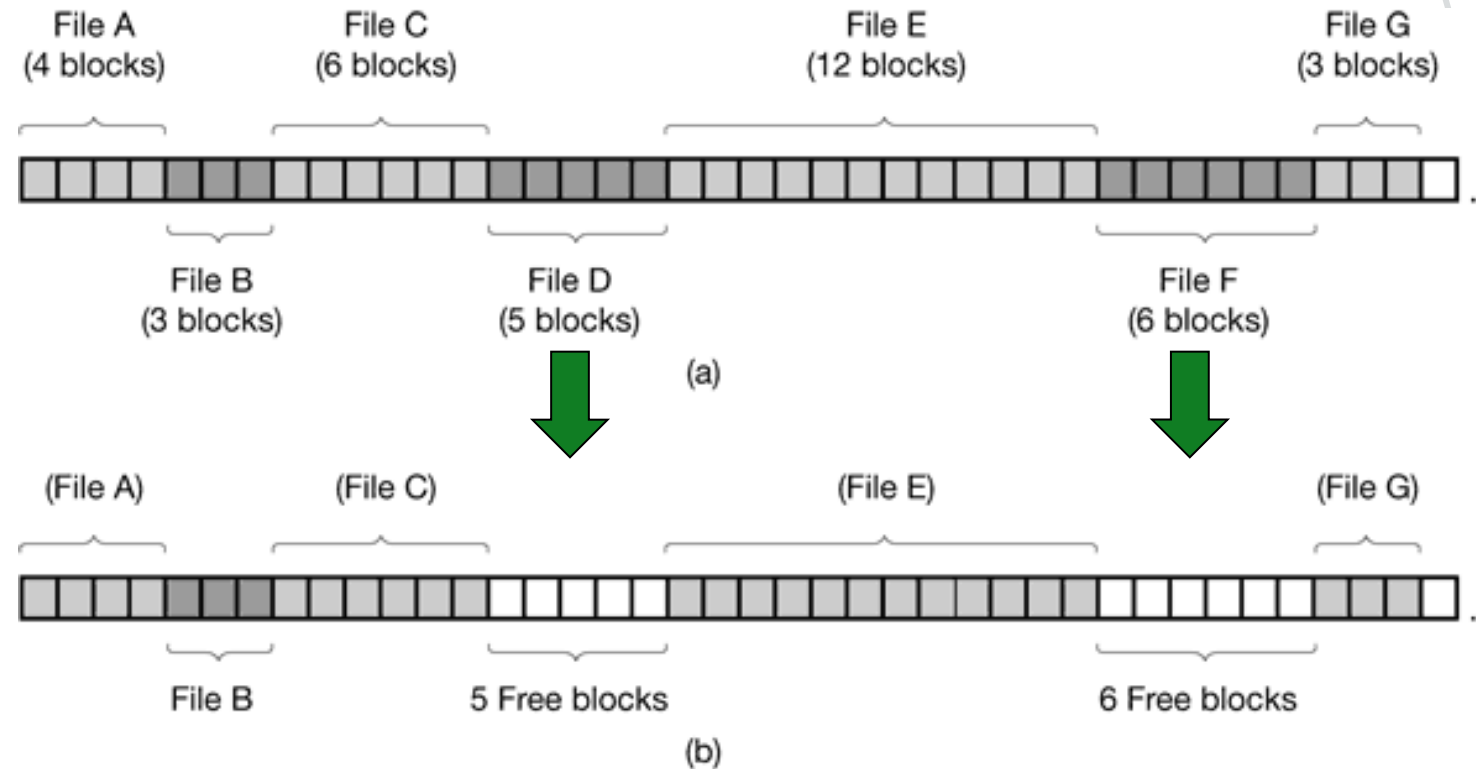
- **Desvantagens:**

- Ao longo do tempo o disco se torna fragmentado - Compactação têm um alto custo.
- Reuso do espaço - atualização da lista de espaços livres
- Problemas com fragmentação nos blocos - conhecimento do tamanho final do arquivo para alocar espaço necessário.



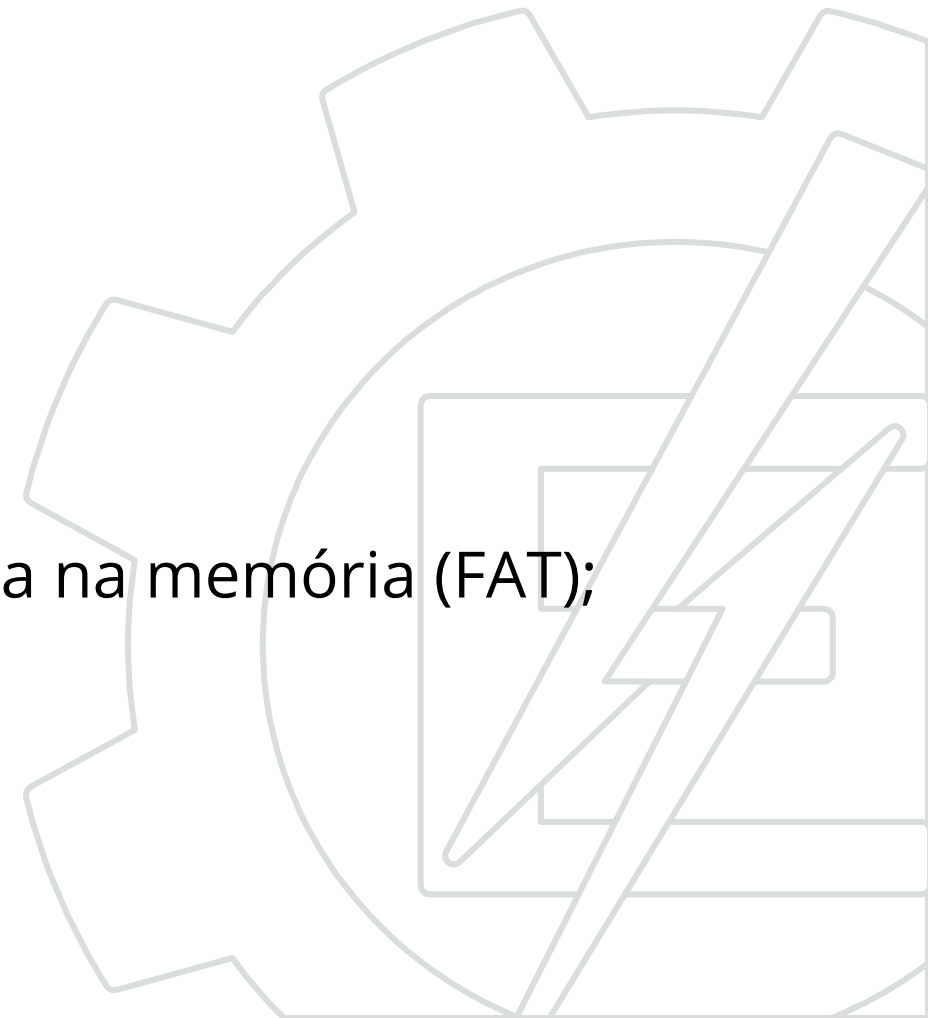
Implementando arquivos - Alocação contínua/contígua

- **Desvantagens:**
 - Com a remoção de D e F surgem buracos:



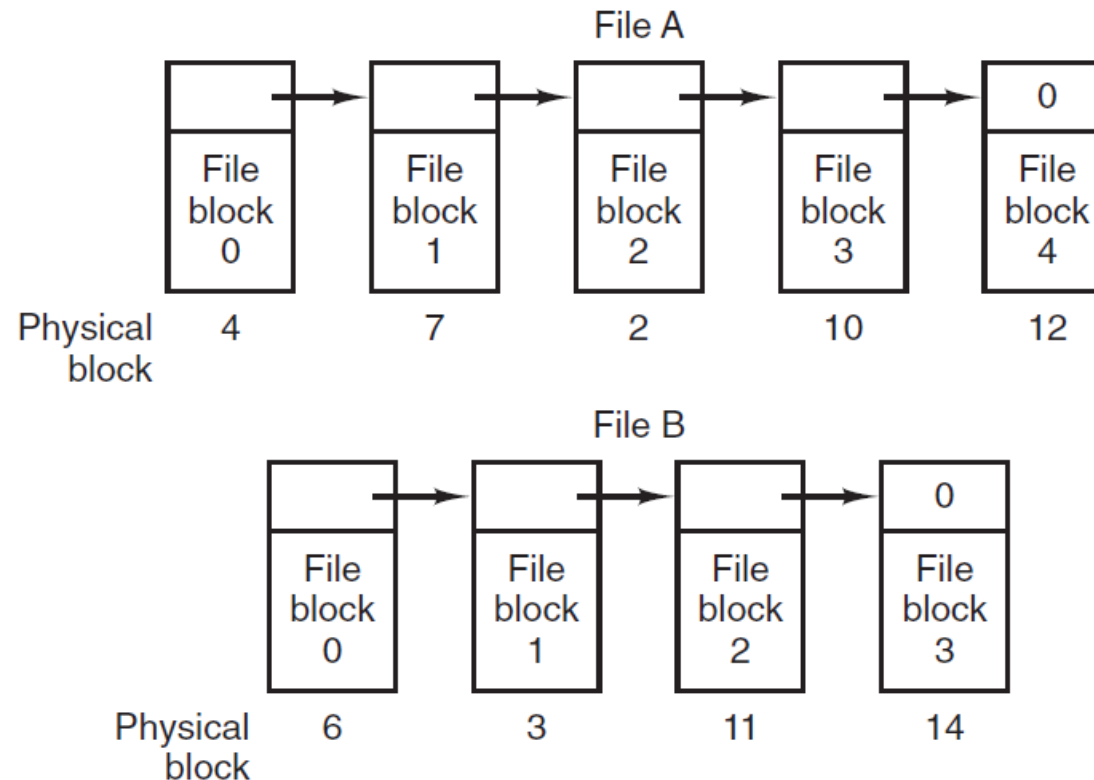
Implementando arquivos

- Como os arquivos são alocados no disco?
- Diferentes técnicas por diferentes SOs:
 - Alocação contínua;
 - **Alocação com lista encadeada;**
 - Alocação com lista enc. utilizando uma tabela na memória (FAT);
 - *i-nodes*.



Alocação com lista encadeada

- Cada arquivo é uma lista ligada de blocos do disco.
- A primeira palavra é o endereço do bloco seguinte e o restante do bloco é destinado aos dados.



Sistema de Arquivos

Implementação

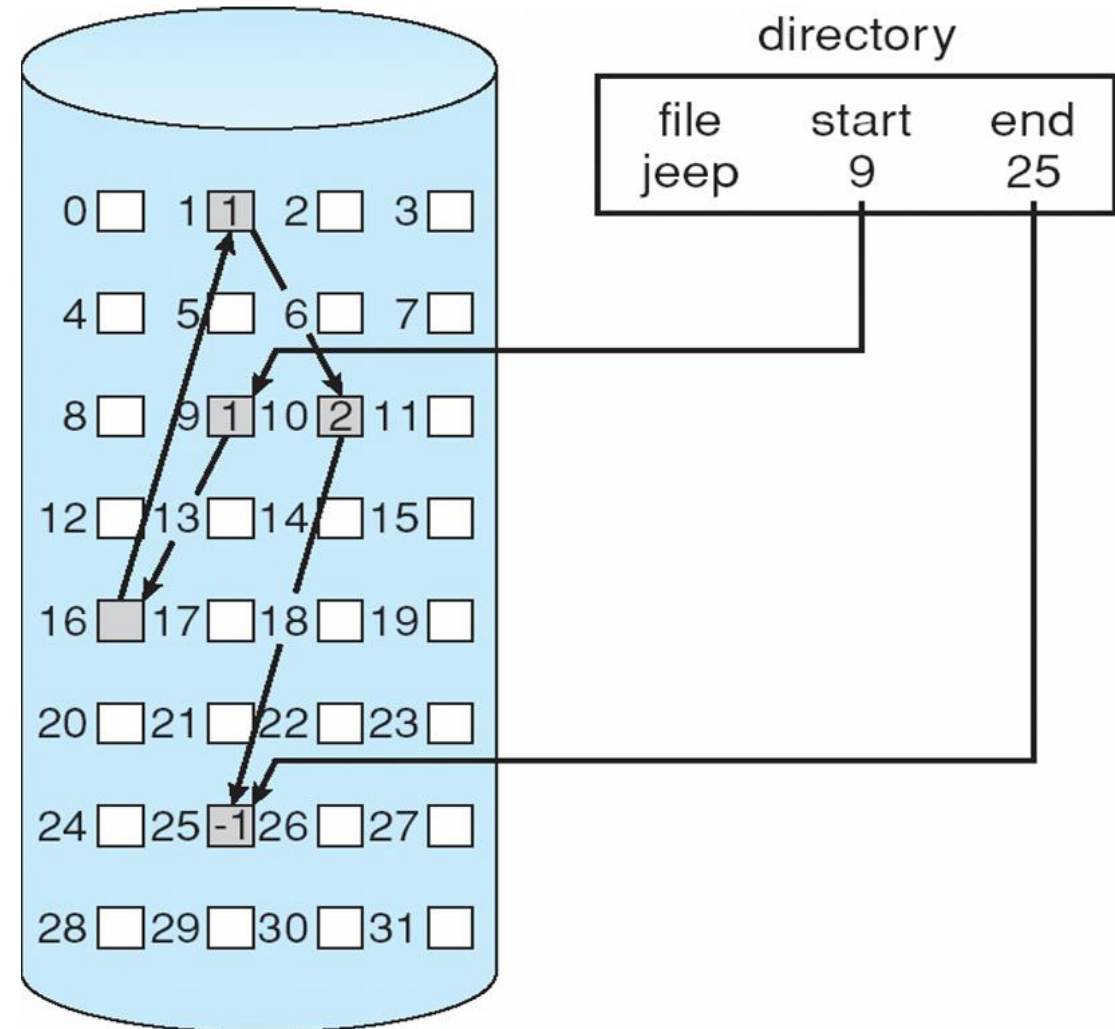
Alocação com lista encadeada

Vantagens:

- Basta que a entrada no diretório armazene o endereço em disco do primeiro bloco.
- Não se perde espaço com fragmentação externa – somente no último bloco.

Desvantagens:

- Acesso aos arquivos é feito sequencialmente – Acesso aleatório é lento.
- É necessário armazenar o ponteiro para o próximo bloco.

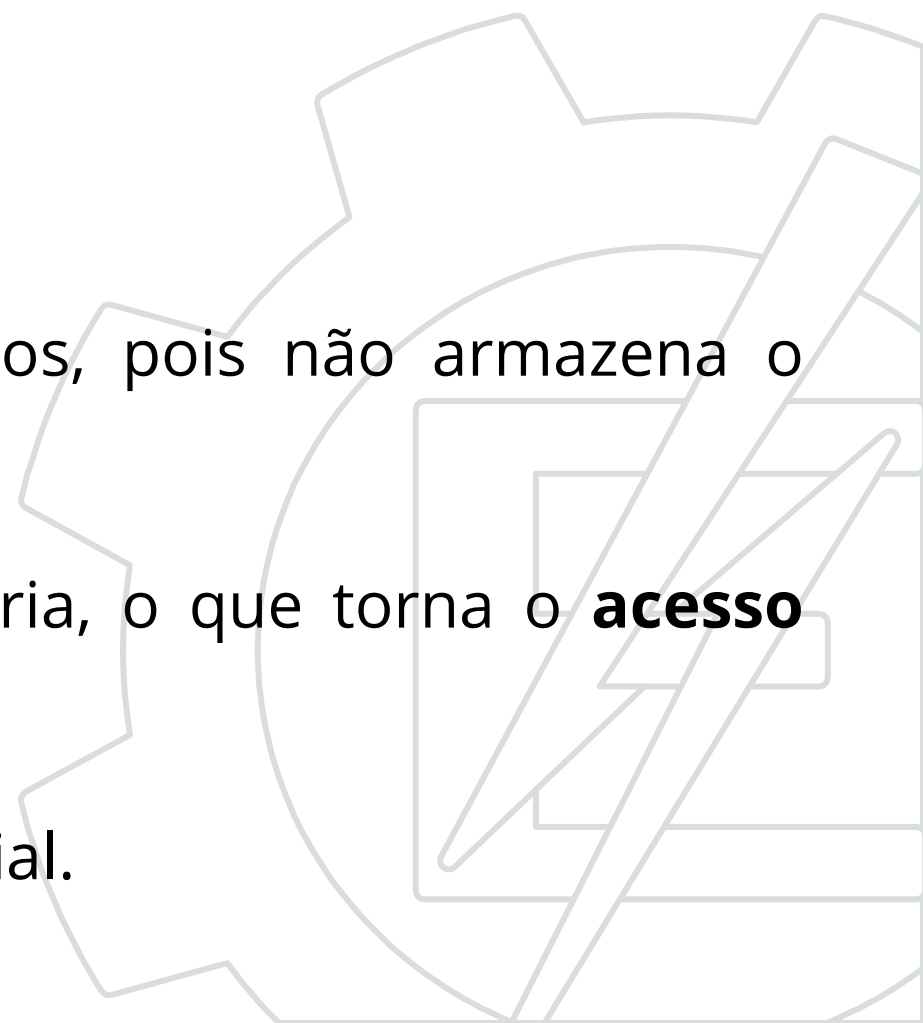


Implementando arquivos

- Como os arquivos são alocados no disco?
- Diferentes técnicas por diferentes SOs:
 - Alocação contínua;
 - Alocação com lista encadeada;
 - **Alocação com lista enc. utilizando uma tabela na memória (FAT);**
 - *i-nodes*.



Alocação com lista encadeada utilizando uma tabela na memória (FAT)

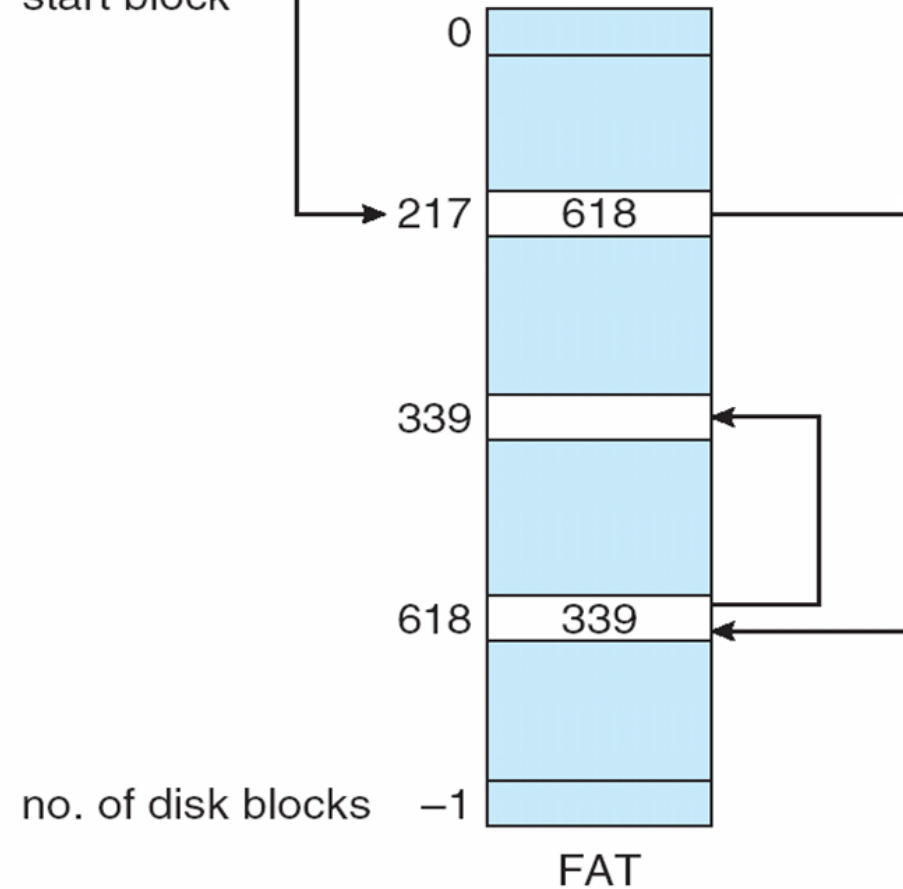
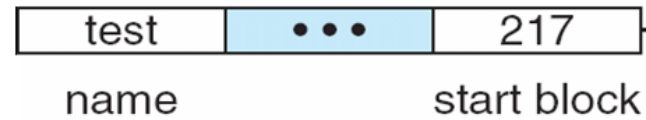
- O ponteiro é colocado em uma tabela na memória, em vez de ser colocado no bloco.
 - FAT (*File Allocation Table*) usa este mecanismo.
 - O bloco inteiro está disponível para os dados, pois não armazena o ponteiro.
 - Toda a cadeia de um arquivo está na memória, o que torna o **acesso aleatório mais rápido**.
 - O Diretório armazena o endereço do bloco inicial.
- 

Sistema de Arquivos

Implementação

Alocação com lista encadeada utilizando uma tabela na memória (FAT)

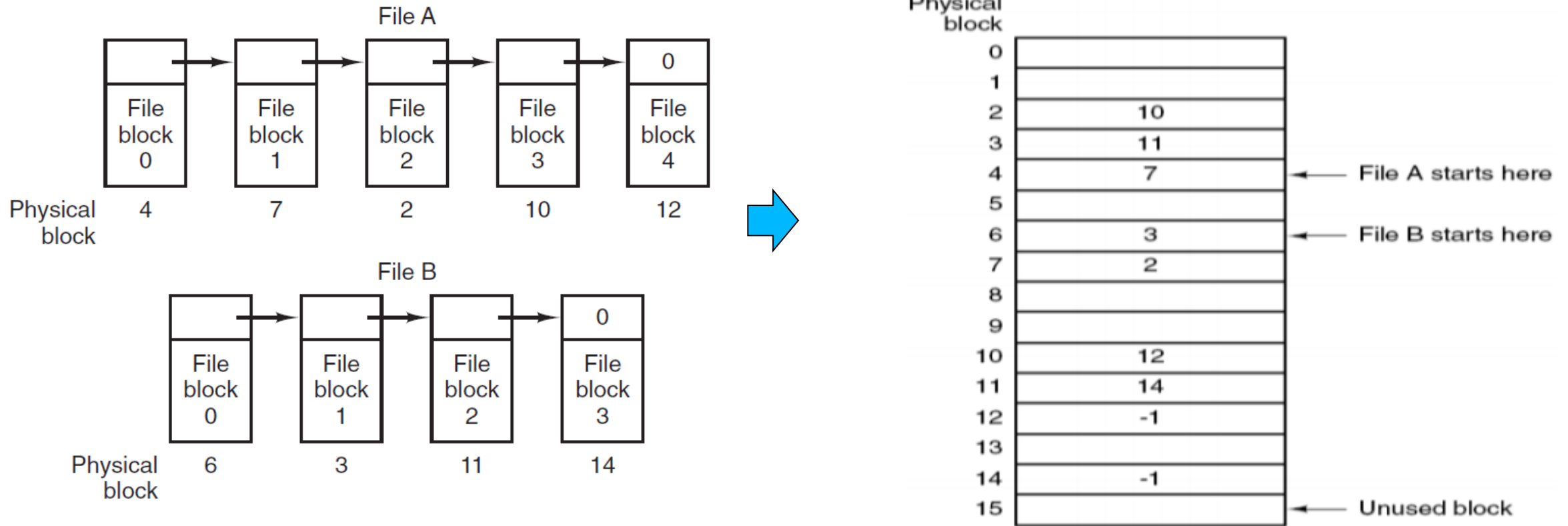
directory entry



Sistema de Arquivos

Implementação

Alocação com lista encadeada utilizando uma tabela na memória (FAT)



Linked list allocation using a file allocation table in main memory.

Alocação com lista encadeada utilizando uma tabela na memória (FAT)

- MS-DOS e Windows 9x utilizam este modo de alocação.
- WinNT, Win2000 e WinXP utilizam NTFS.
- **Desvantagens:**
 - Toda a tabela deve estar na memória e ocupa muito espaço.
 - Exemplo - Disco de 200 GB com blocos de 1 KB:
 - A tabela precisa de 200 milhões de entradas, cada uma com 4 bytes, o que equivale a 800MB de memória.

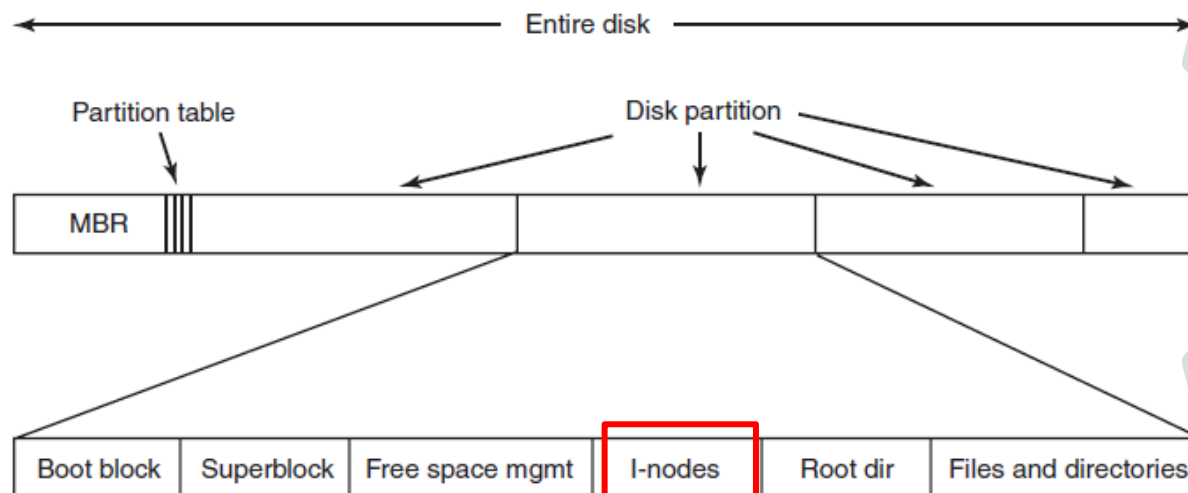
Implementando arquivos

- Como os arquivos são alocados no disco?
- Diferentes técnicas por diferentes SOs:
 - Alocação contínua;
 - Alocação com lista encadeada;
 - Alocação com lista enc. utilizando uma tabela na memória (FAT);
 - ***i-nodes***.



i-Nodes

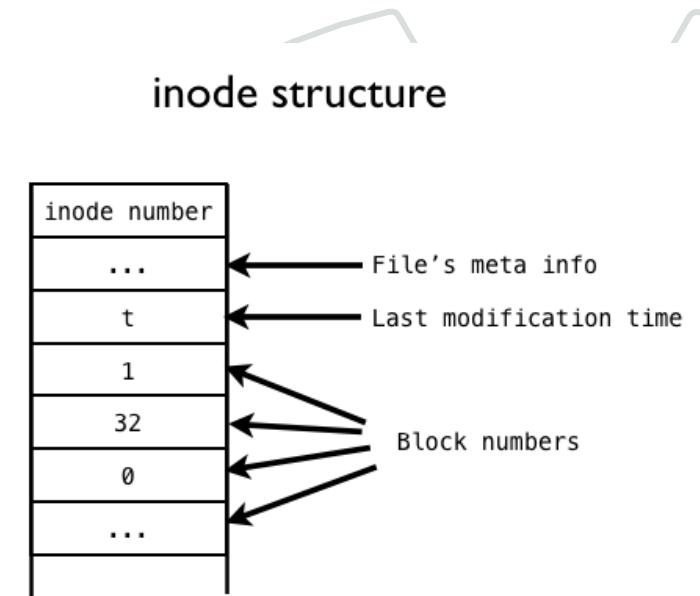
- Utilizados nos Sistemas Operacionais Unix e Linux.
- Cada arquivo possui uma estrutura chamada ***i-node***.
- O ***i-node*** contém atributos e endereços no disco dos blocos do arquivo.
- A partir do ***i-node*** de um arquivo, é possível encontrar todos os blocos desse arquivo.



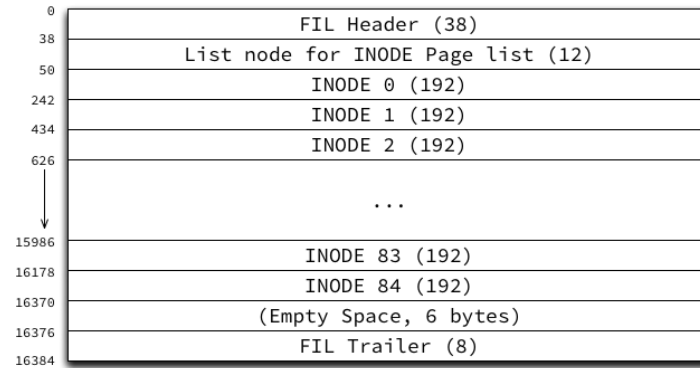
i-Nodes

Vantagens:

- O ***i-node*** só é carregado na memória quando o arquivo é aberto.
- O espaço de memória ocupado pelos ***i-nodes*** é proporcional ao número de arquivos que podem ser abertos ao mesmo tempo.
- O espaço de memória ocupado pela FAT é proporcional ao tamanho do disco.



INODE Overview



Inode Structure (I)

Bytes	Description
0-1	File mode (type & permissions)
2-3	UID lower 2 bytes
4-7	Size lower 4 bytes
8-11	Access Time
12-15	Change Time
16-19	Modification Time
20-23	Delete Time
24-25	GID lower 2 bytes
26-27	Link count
28-31	Sector count
32-35	Flags
36-39	unused

Inode Structure (II)

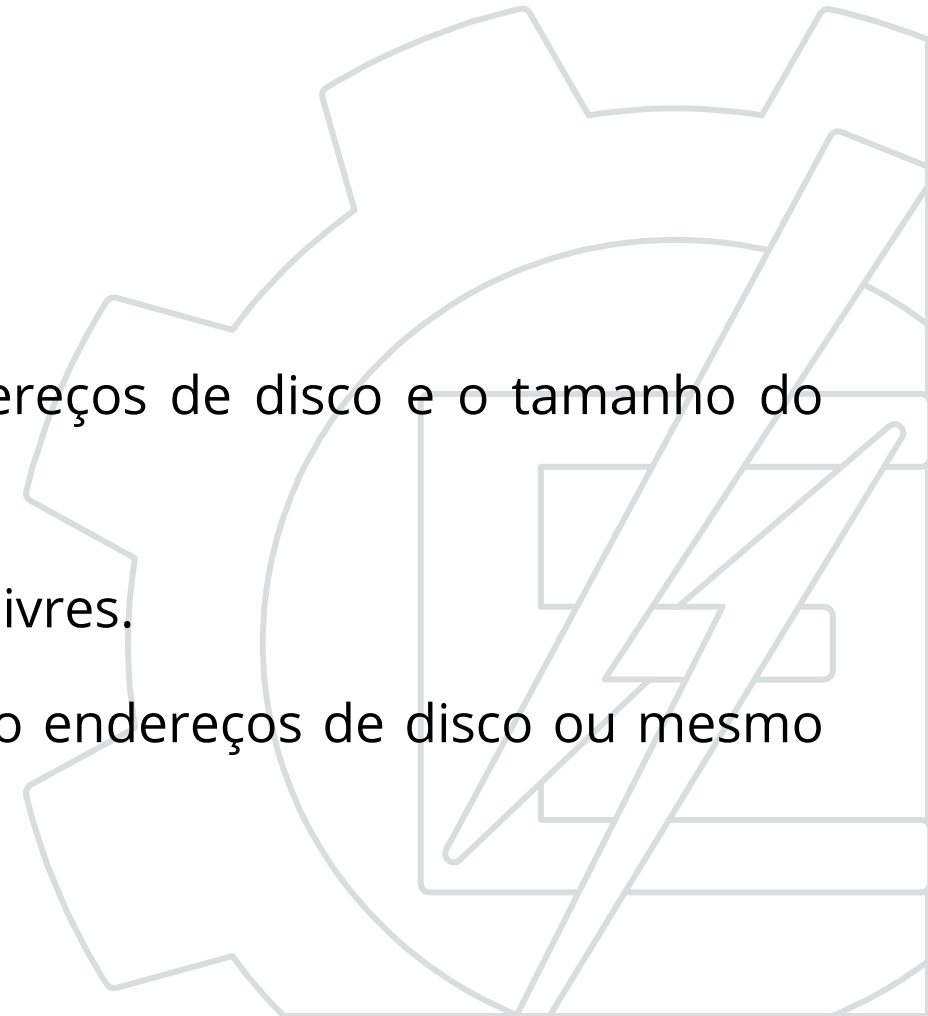
Bytes	Description
40-87	12 direct block pointers
88-91	1 single-indirect pointer
92-95	1 double-indirect pointer
96-99	1 triple-indirect pointer
100-103	Generation number
104-107	Extended attribute block address
108-111	Size upper 4 bytes OR Directory ACL
112-115	Fragment block address
116-117	Fragment size
118-119	unused
120-121	UID upper 2 bytes
122-123	GID upper 2 bytes
124-124	unused

i-Nodes

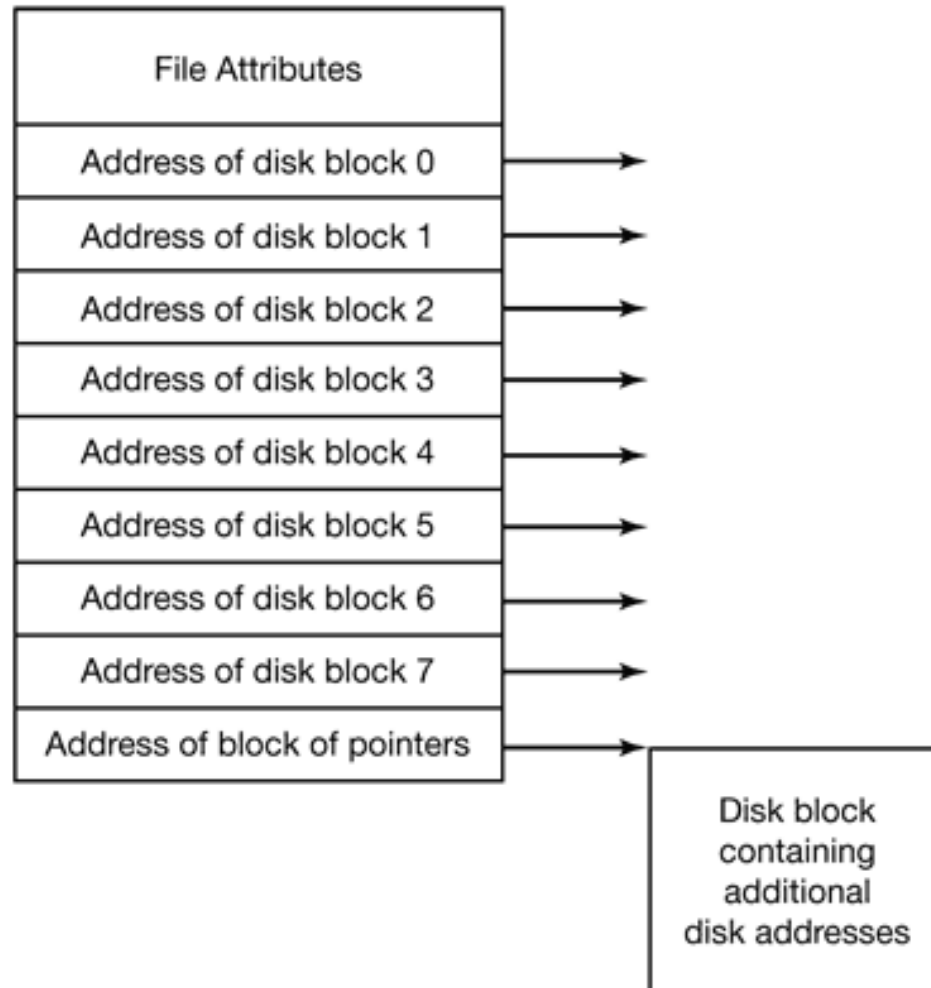
- Se cada ***i-node*** ocupa ***n bytes*** e é possível que se tenha no máximo ***k arquivos*** abertos, então o total de memória ocupada é ***k*n bytes***.

Desvantagens:

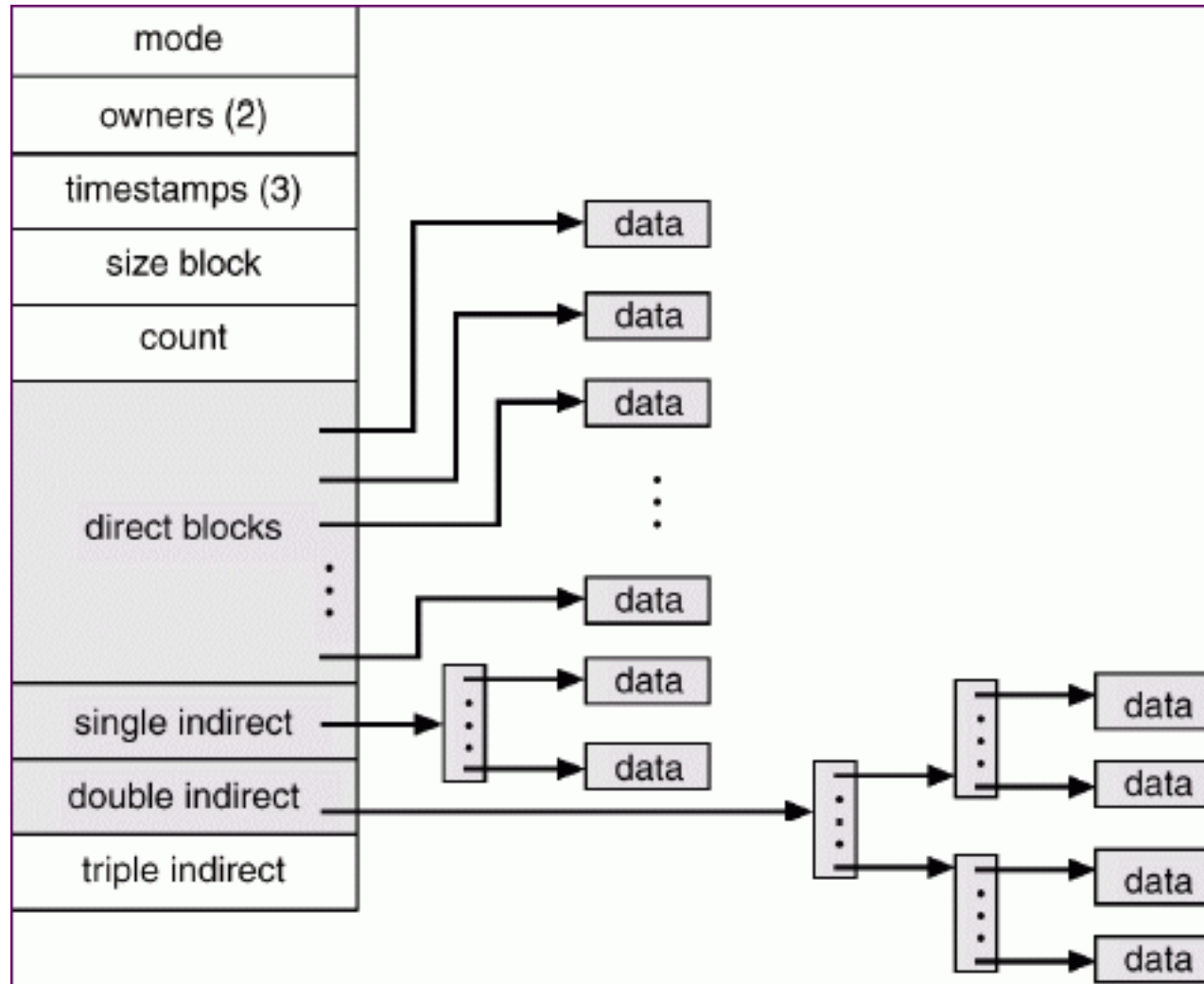
- Cada ***i-node*** tem espaço para um número fixo de endereços de disco e o tamanho do arquivo pode aumentar além deste limite.
- **Solução:** reservar o último ***i-node*** para blocos com mais livres.
 - Pode-se ter até dois ou mais desses blocos contendo endereços de disco ou mesmo blocos apontando a outros blocos de endereços.



i-Nodes



i-Nodes



i-Nodes - /usr/ast/mbox

Root directory

1	.
1	..
4	bin
7	dev
14	lib
9	etc
6	usr
8	tmp

Looking up
usr yields
i-node 6

I-node 6
is for /usr

Mode size times
132

I-node 6
says that
/usr is in
block 132

Block 132
is /usr
directory

6	.
1	..
19	dick
30	erik
51	jim
26	ast
45	bal

/usr/ast
is i-node
26

I-node 26
is for
/usr/ast

Mode size times
406

I-node 26
says that
/usr/ast is in
block 406

Block 406
is /usr/ast
directory

26	.
6	..
64	grants
92	books
60	mbox
81	minix
17	src

/usr/ast/mbox
is i-node
60

Sistema de Arquivos

Implementação

i-Nodes - */usr/andy/Texto.txt*



Sistema de Arquivos

Implementação

i-Nodes - */usr/andy/Texto.txt*

Diretório Raiz

1	.
1	..
4	bin
6	usr
7	tmp

Encontra */usr*
com *i-node* 6



Sistema de Arquivos

Implementação

i-Nodes - */usr/andy/Texto.txt*

Diretório Raiz *

1	.
1	..
4	bin
6	usr
7	tmp

Encontra */usr*
com *i-node* 6



I-node 6

modo
tamanho
hora
132



Bloco 132 tem
o sub-diretório
/usr



Sistema de Arquivos

Implementação

i-Nodes - */usr/andy/Texto.txt*

Diretório Raiz *

1 .
1 ..
4 bin
6 usr
7 tmp

Encontra */usr*
com *i-node* 6



I-node 6

modo
tamanho
hora
132

Bloco 132 tem
o sub-diretório
/usr



Bloco 132

6 .
1 ..
19 dick
23 ana
26 andy

/usr/andy está no
i-node 26



Sistema de Arquivos

Implementação

i-Nodes - */usr/andy/Texto.txt*

Diretório Raiz *

1 .
1 ..
4 bin
6 usr
7 tmp

Encontra */usr*
com *i-node 6*



I-node 6

modo
tamanho
hora
132

Bloco 132 tem
o sub-diretório
/usr



Bloco 132

6 .
1 ..
19 dick
23 ana
26 andy

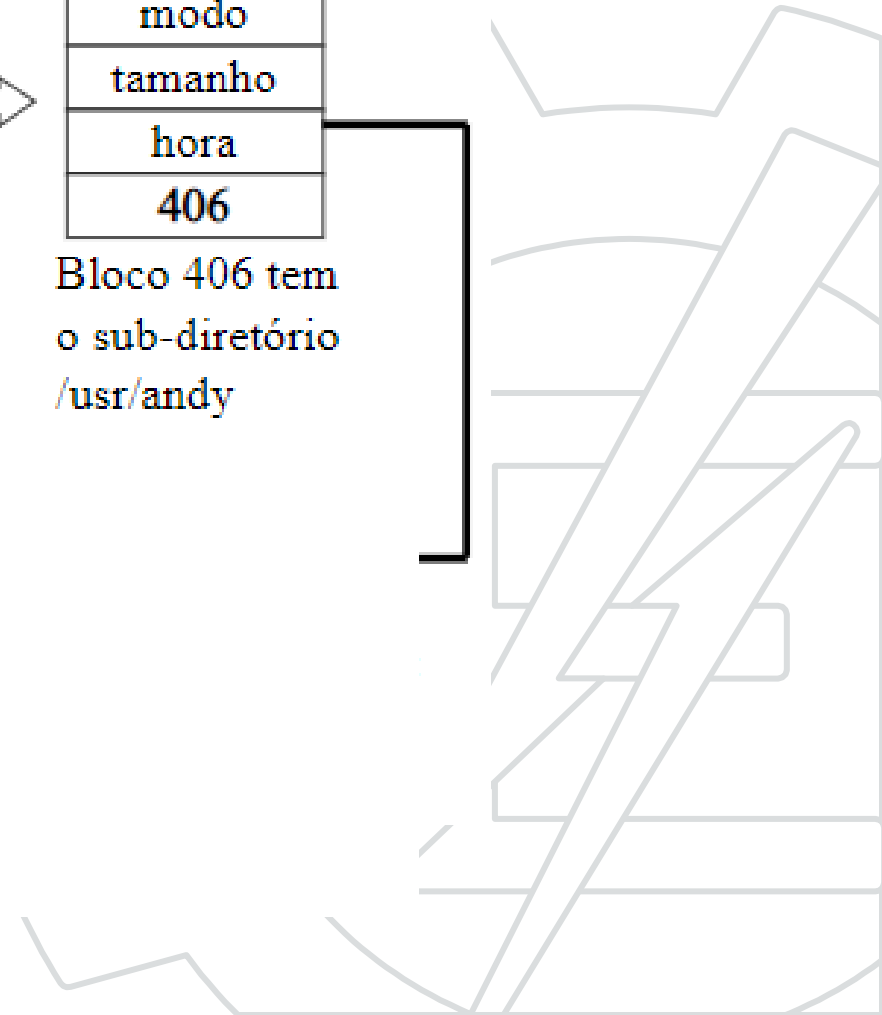
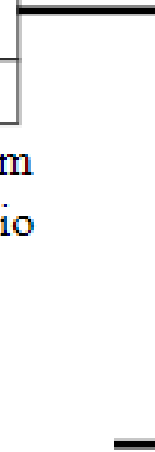
/usr/andy está no
i-node 26



I-node 26

modo
tamanho
hora
406

Bloco 406 tem
o sub-diretório
/usr/andy



Sistema de Arquivos

Implementação

i-Nodes - `/usr/andy/Texto.txt`

Diretório Raiz *

1 .
1 ..
4 bin
6 usr
7 tmp

Encontra `/usr`
com *i-node* 6

I-node 6

modo
tamanho
hora
132

Bloco 132 tem
o sub-diretório
`/usr`

Bloco 132

6 .
1 ..
19 dick
23 ana
26 andy

`/usr/andy` está no
i-node 26

I-node 26

modo
tamanho
hora
406

Bloco 406 tem
o sub-diretório
`/usr/andy`

Bloco 406

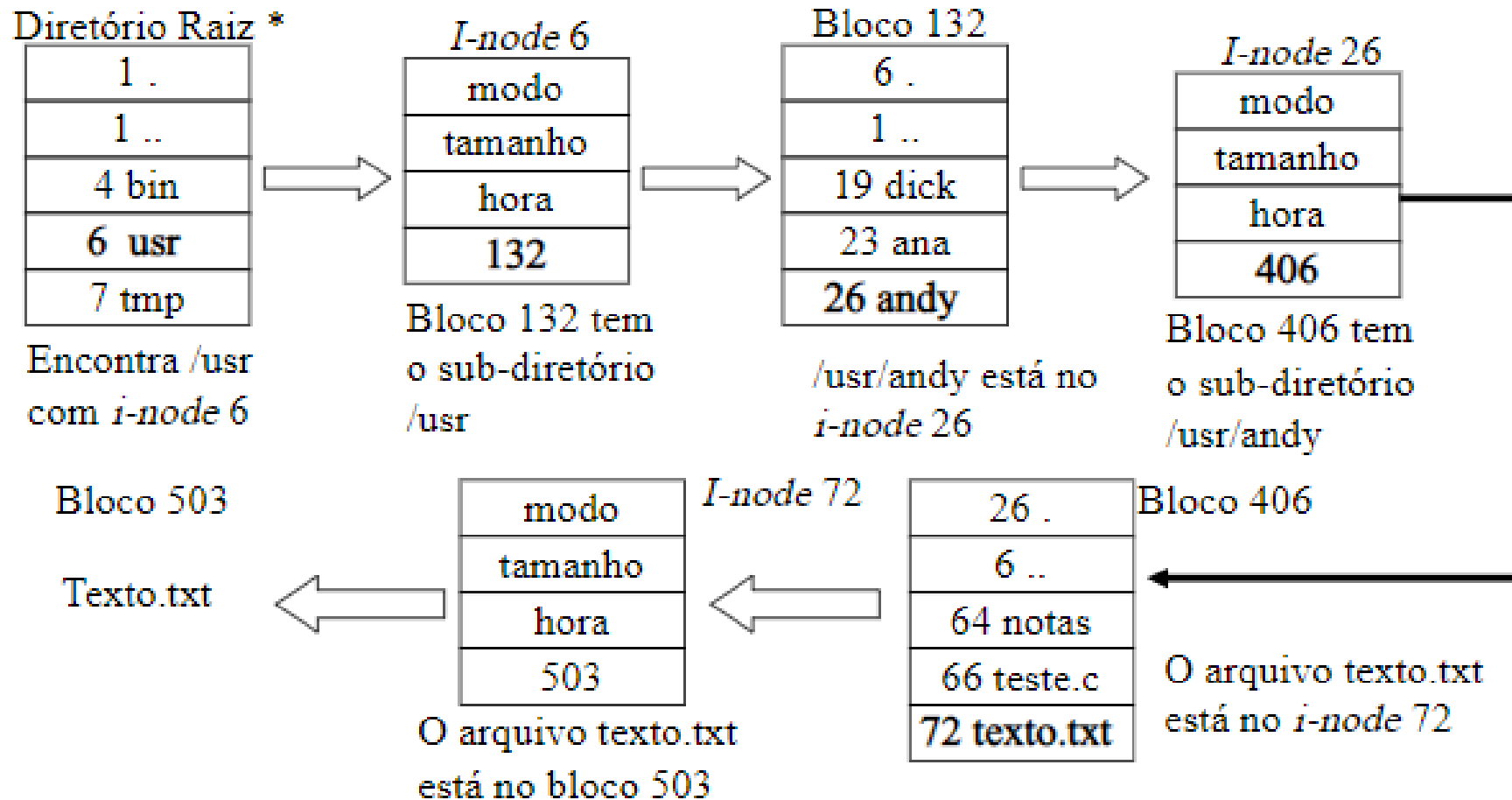
26 .
6 ..
64 notas
66 teste.c
72 texto.txt

O arquivo `texto.txt`
está no *i-node* 72

Sistema de Arquivos

Implementação

i-Nodes - `/usr/andy/Texto.txt`



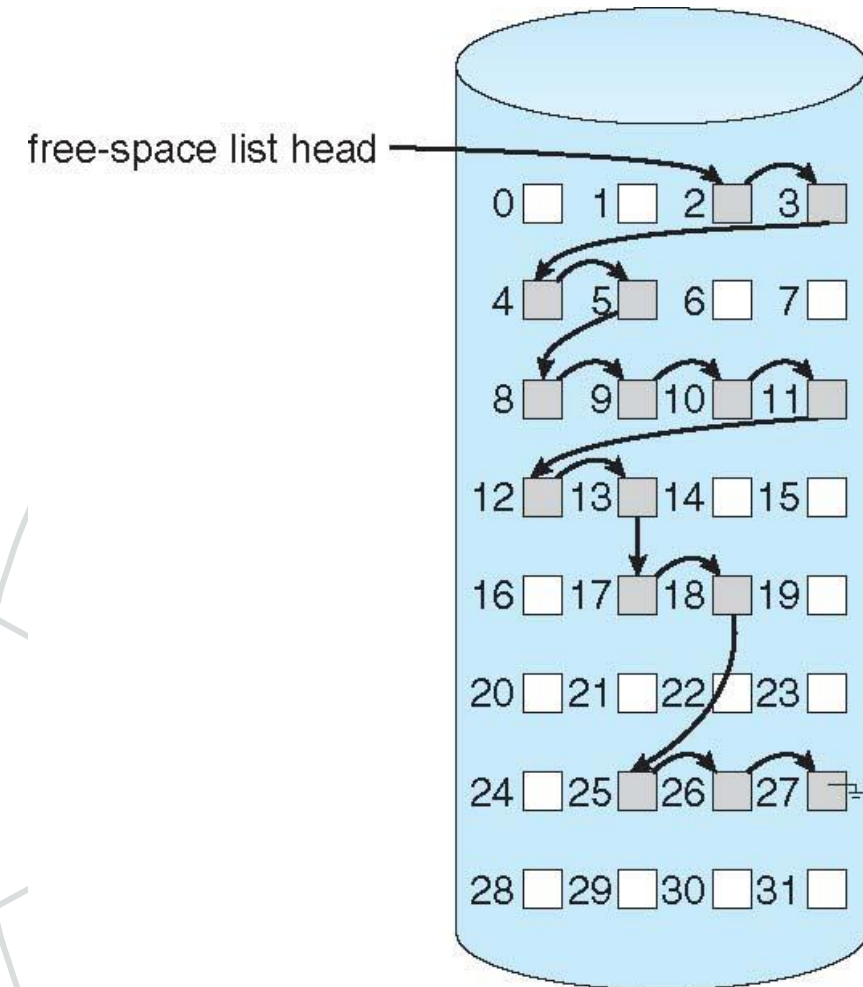
Sistema de Arquivos

Gerenciamento de blocos livres



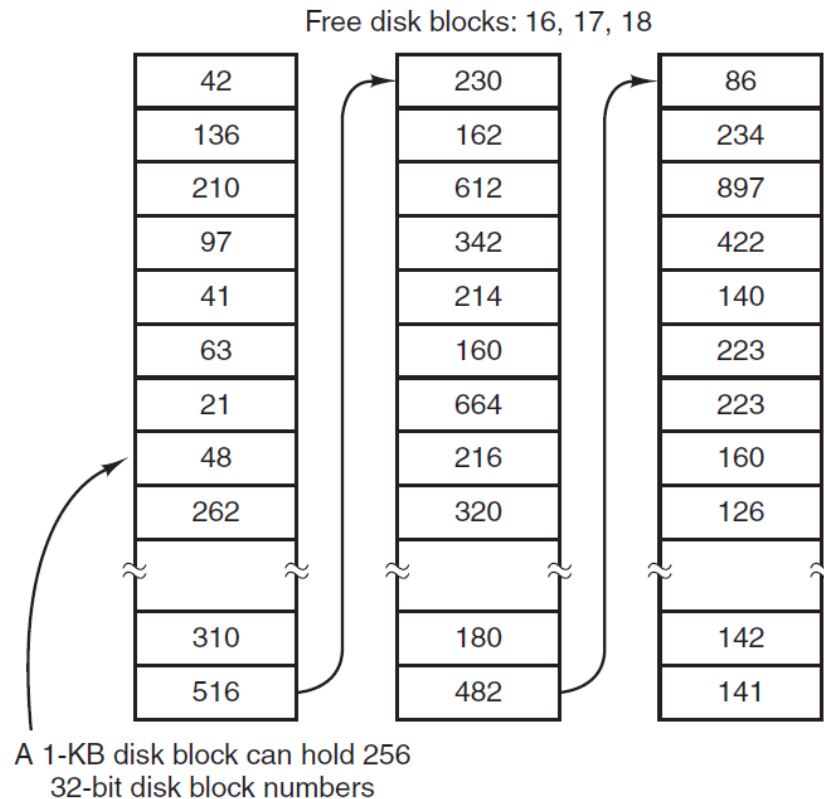
1a) Lista ligada de blocos de disco

- Um ponteiro para o primeiro bloco livre mantido em um local específico do disco e em cache (memória).
- Este primeiro bloco contém o endereço do segundo, e assim por diante.
- **Problema:** pouco eficiente caso seja necessário varrer a lista.



1b) Lista ligada de blocos de disco - alternativa

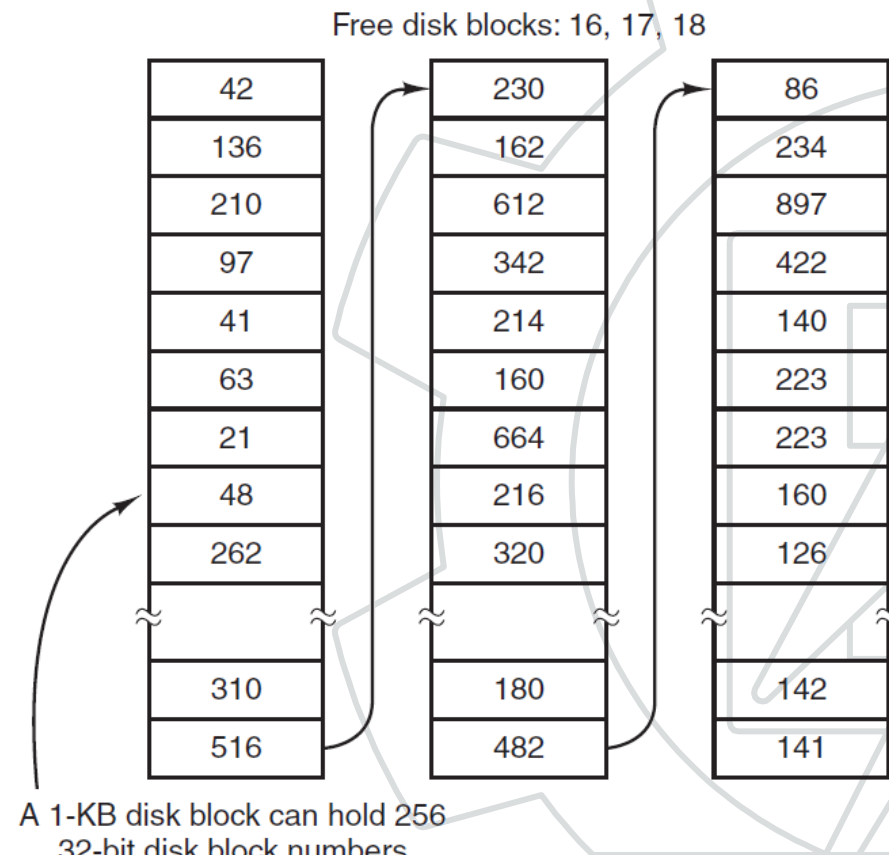
- Armazenar no primeiro bloco o endereço de n-1 blocos livres.
- Guardar o ponteiro do próximo bloco de endereços livres na última posição.



Sistema de Arquivos

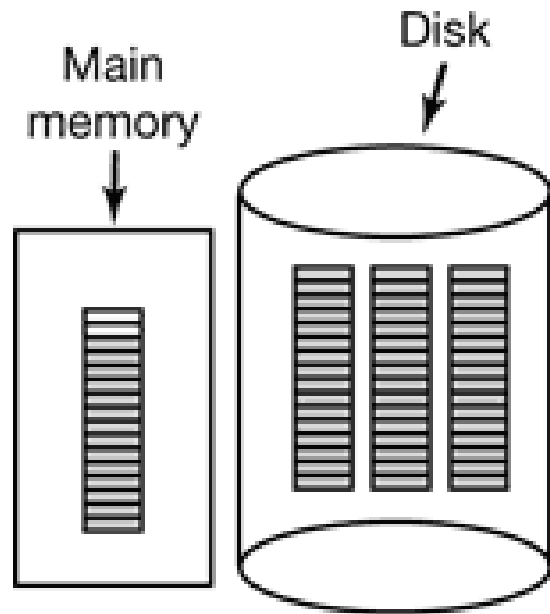
Gerenciamento de blocos livres

- Somente um bloco de ponteiros na memória principal.
- Quando o arquivo é criado, os blocos livres são retirados.
- Quando o bloco se esgotar, um novo bloco é lido do disco.
- Quando o arquivo é apagado, os blocos são liberados e adicionados ao bloco de ponteiros na memória.
- Quando o bloco está completo ele é escrito no disco.

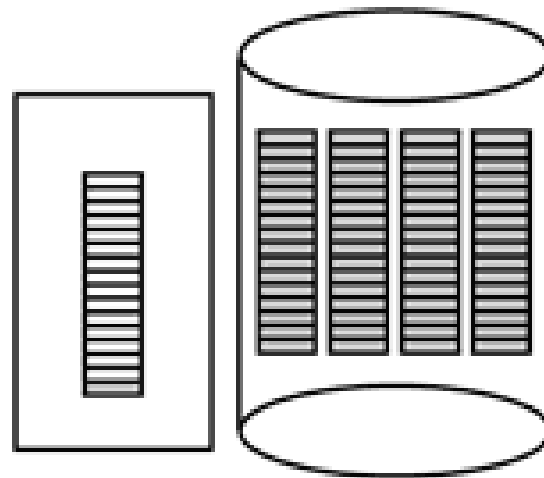


Problema com arquivos temporários

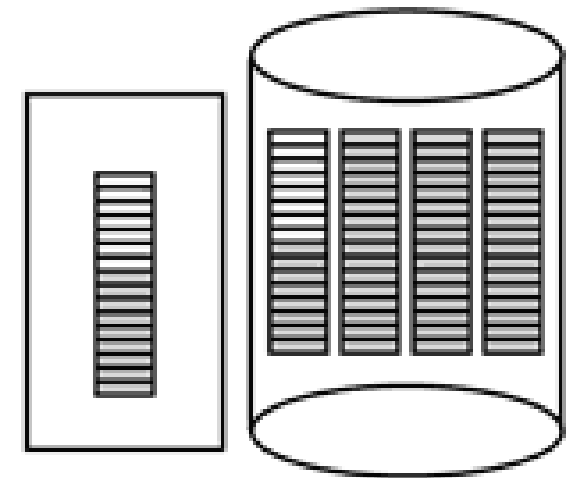
- Muita E/S com a criação e exclusão de arquivos temporários.
- Dividir a tabela de blocos livres?



(a)



(b)



(c)

Lista ligada de blocos de disco

Vantagens

- Requer menos espaço se o disco está quase cheio;
- Armazena apenas um bloco de ponteiros na memória.

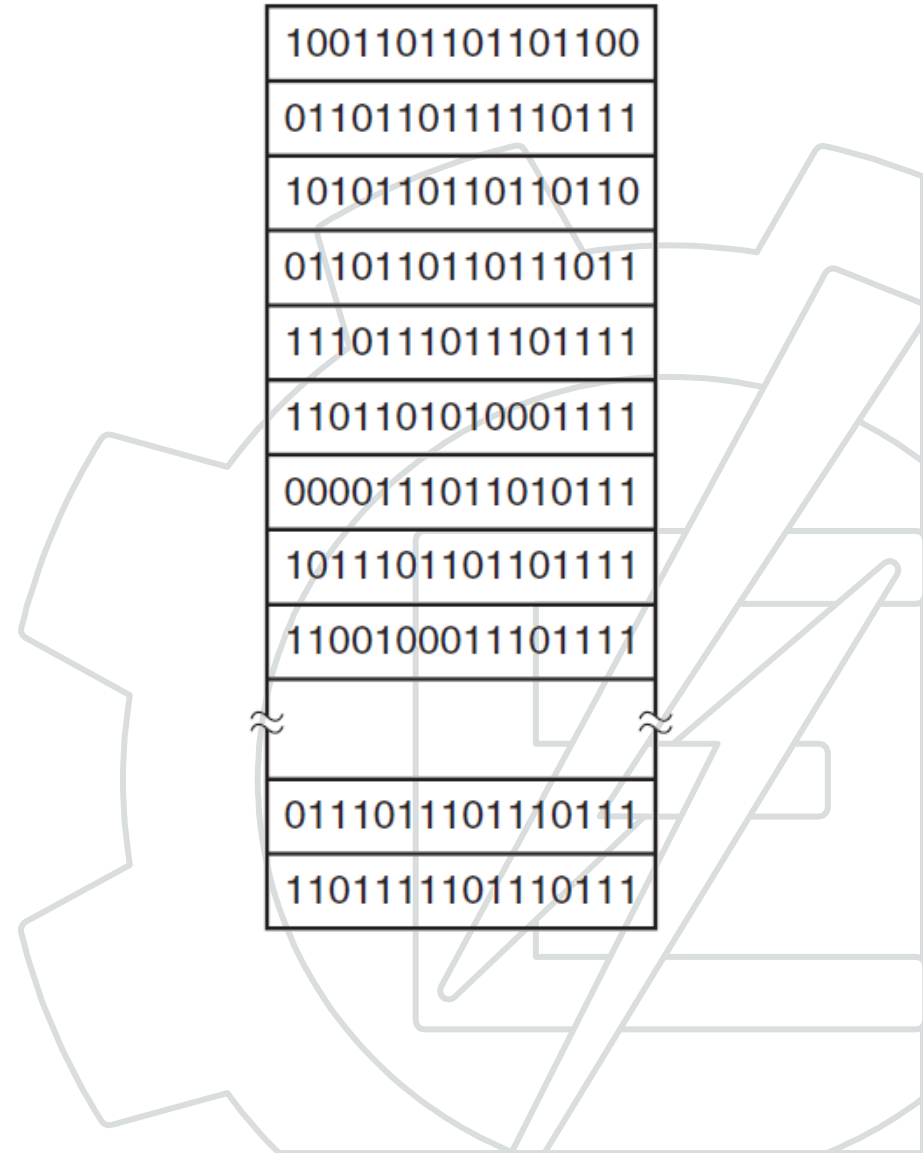
Desvantagens

- Requer mais espaço se o disco está vazio;
- Dificulta a alocação contígua.



2) Mapa de bits

- Depende do tamanho do disco – quanto maior o disco, maior o mapa.
- Um disco com n blocos requer um mapa com n bits, sendo um bit para cada bloco.
- O mapa é mantido na memória principal.
- **Vantagem:** facilita a alocação contígua.
- **Desvantagem:** torna-se lento quando o disco está cheio, pois é necessário varrer boa parte do mapa de bits.

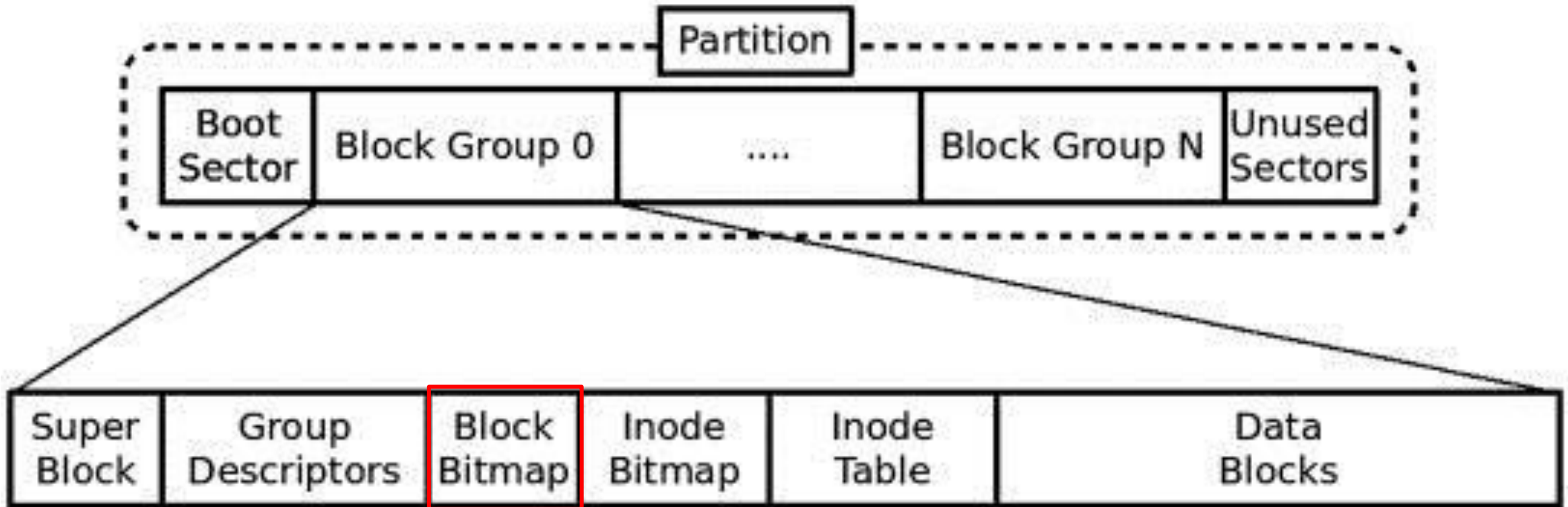


The diagram shows a vertical stack of 14 rectangular boxes, each containing a 16-bit binary string. The boxes are arranged in a column, with a large gear-like shape in the background. The first 10 boxes are connected by a line, and the last 4 boxes are connected by another line, with a break symbol (two wavy lines) indicating a gap between the two groups. The binary strings are as follows:

1001101101101100
0110110111110111
1010110110110110
0110110110111011
1110111011101111
1101101010001111
0000111011010111
1011101101101111
1100100011101111
⋮
0111011101110111
1101111101110111

Sistema de Arquivos

Gerenciamento de blocos livres

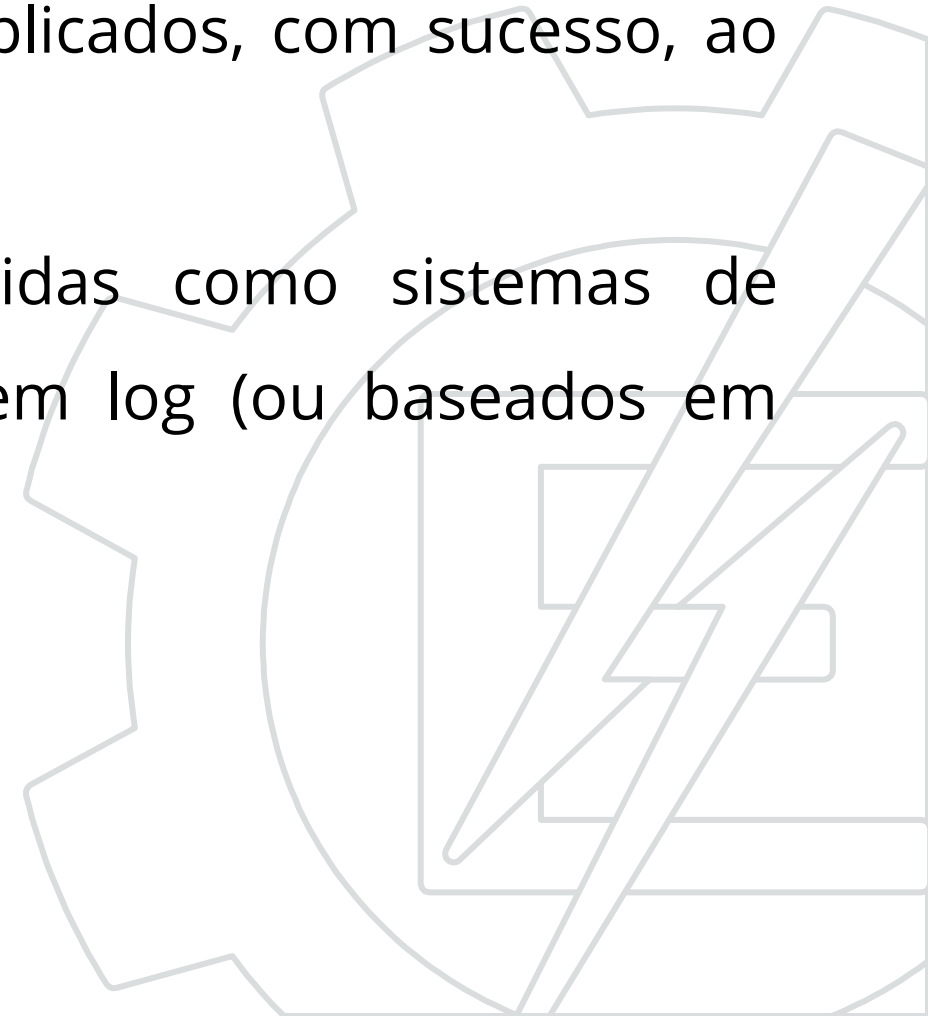


Verificação de Consistência

- Qualquer que seja a causa da corrupção, um sistema de arquivos deve primeiro detectar os problemas e, então, corrigí-los. Para detecção, uma varredura de todos os metadados em cada sistema de arquivos pode confirmar ou negar a consistência do sistema.
- O verificador de consistência — um programa de sistema como o *fsck* no UNIX — compara os dados na estrutura de diretórios com os blocos de dados em disco e tenta corrigir qualquer inconsistência encontrada.
- Os algoritmos de alocação e de gerenciamento do espaço livre definem que tipos de problemas o verificador pode encontrar e o nível de sucesso que ele terá ao corrigí-los.

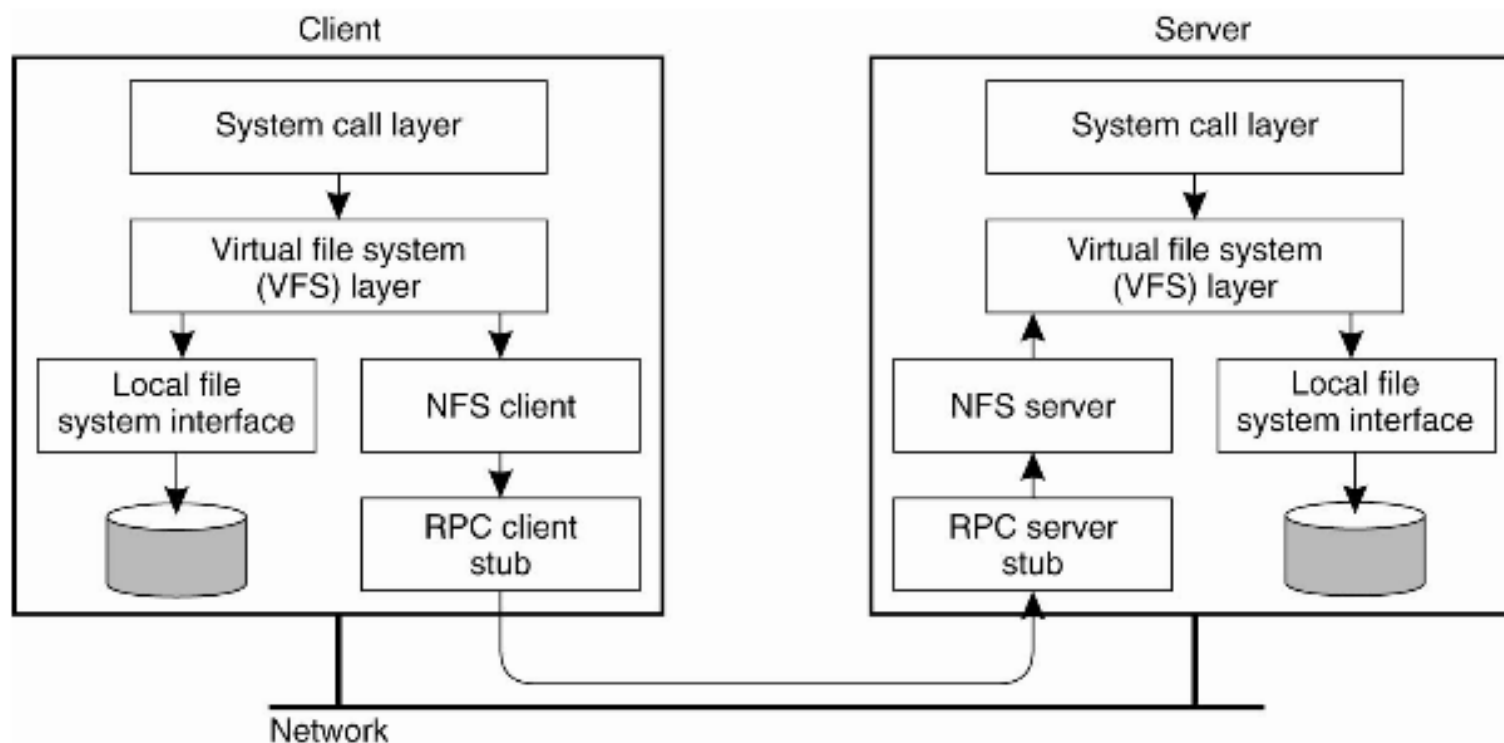
Sistemas de Arquivos Estruturados em Log

- Esses algoritmos baseados em *log* têm sido aplicados, com sucesso, ao problema da verificação de consistência.
- As implementações resultantes são conhecidas como sistemas de arquivos orientados a transações baseados em log (ou baseados em diário - *journaling*).



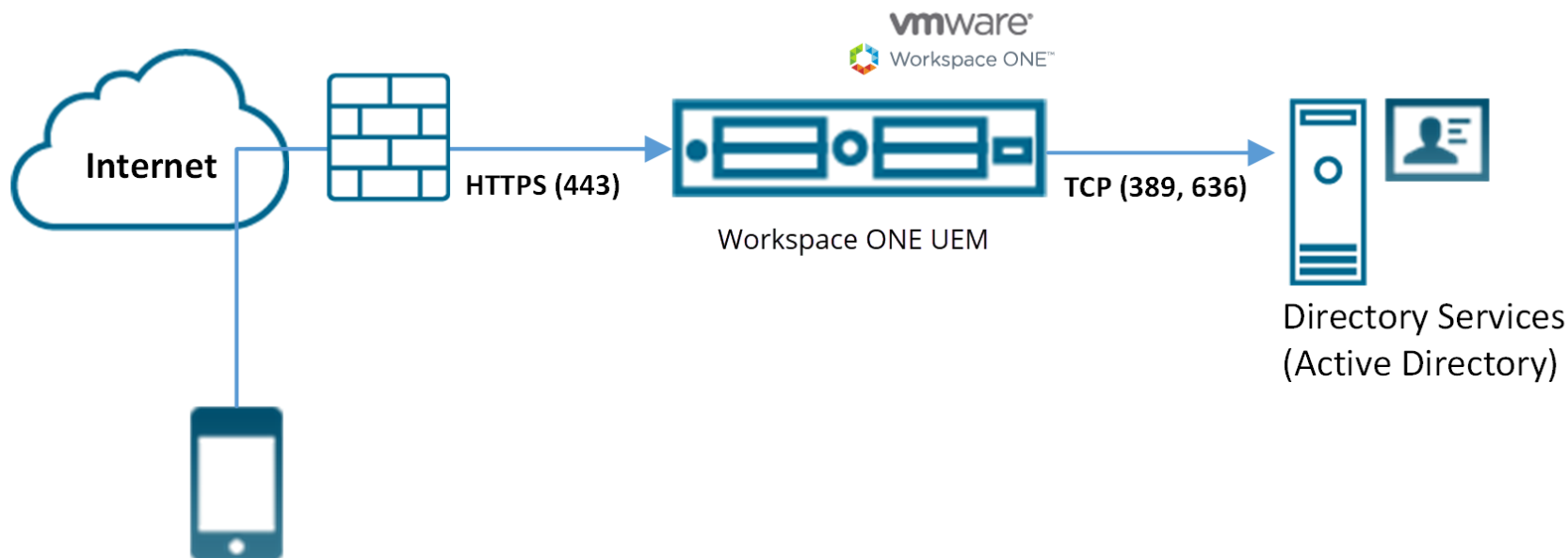
NFS (Network File System)

- Os sistemas de arquivo em rede são comuns. Normalmente, eles são integrados à estrutura geral de diretórios e à interface do sistema cliente. O NFS é um bom exemplo de sistema de arquivos em rede cliente-servidor amplamente usado e bem implementado.



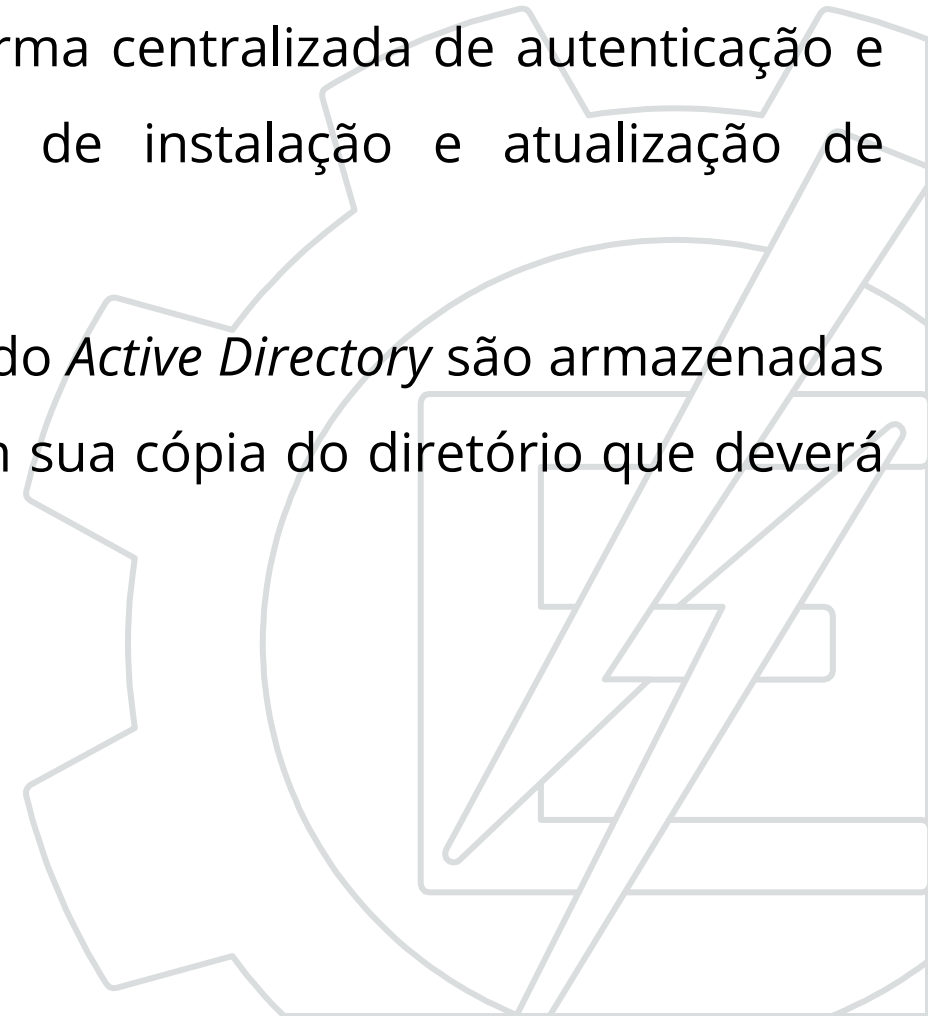
LDAP (*Lightweight Directory Access Protocol*)

- Um LDAP é composto por objetos abstratos que representam recursos (computadores, impressoras, arquivos, etc.) e usuários de uma rede de computadores. Dentre outras operações ao se conectar a um servidor LDAP é possível 1) fazer buscas no diretório; 2) adicionar, modificar e apagar objetos; 3) requerer canal de comunicação seguro.



Active Directory

- É baseado no LDAP e tem por objetivo fornecer uma forma centralizada de autenticação e autorização de serviços. Ele permite definir políticas de instalação e atualização de programas.
- Os usuários são registrados no Domínio. As informações do *Active Directory* são armazenadas em um ou mais Controladores de Domínio, cada um com sua cópia do diretório que deverá ser sincronizada em caso de alteração.



Bibliografia

- TANENBAUM, Andrew S; BOS, Herbert. Sistemas operacionais modernos. 4a ed. São Paulo: Pearson Education do Brasil, 2016.

Capítulo 4.

<https://plataforma.bvirtual.com.br/Acervo/Publicacao/1233>

- DEITEL, H.M; DEITEL, P.J; CHOFFNES,D.R. Sistemas Operacionais. 3a ed. São Paulo: Pearson Prentice Hall, 2005. **Capítulos 12-13.**

<https://plataforma.bvirtual.com.br/Acervo/Publicacao/315>



Sistemas Operacionais

Prof. Otávio Gomes

otavio.gomes@unifei.edu.br

