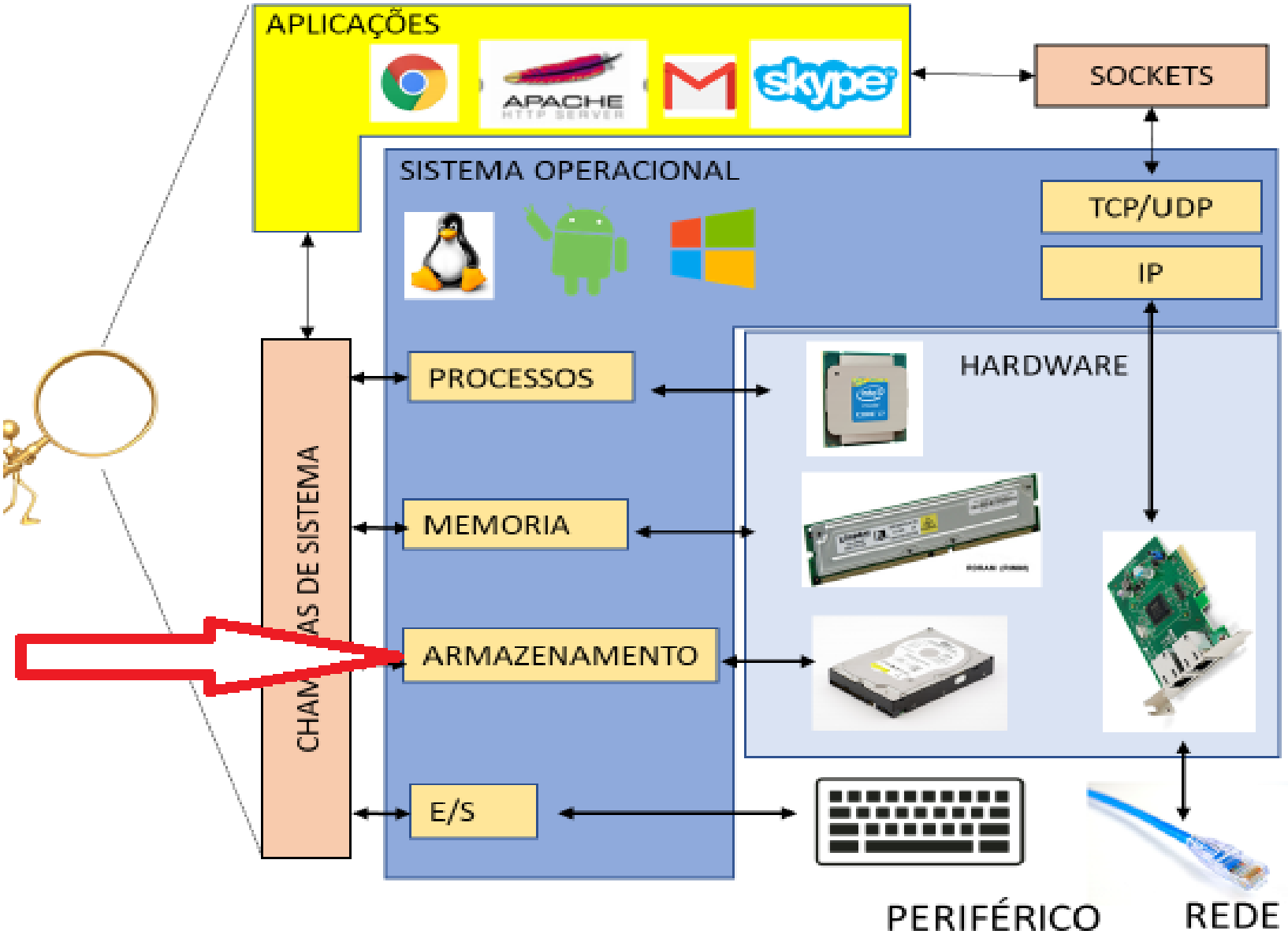
Sistemas de Arquivos


APLICAÇÕES
APACHE
HTTP SERVER
skype
SOCKETS
SISTEMA OPERACIONAL
TCP/UDP
IP
HARDWARE
PROCESSOS
MEMORIA
ARMAZENAMENTO
E/S
CHAMADAS DE SISTEMA
PERIFÉRICO
REDE

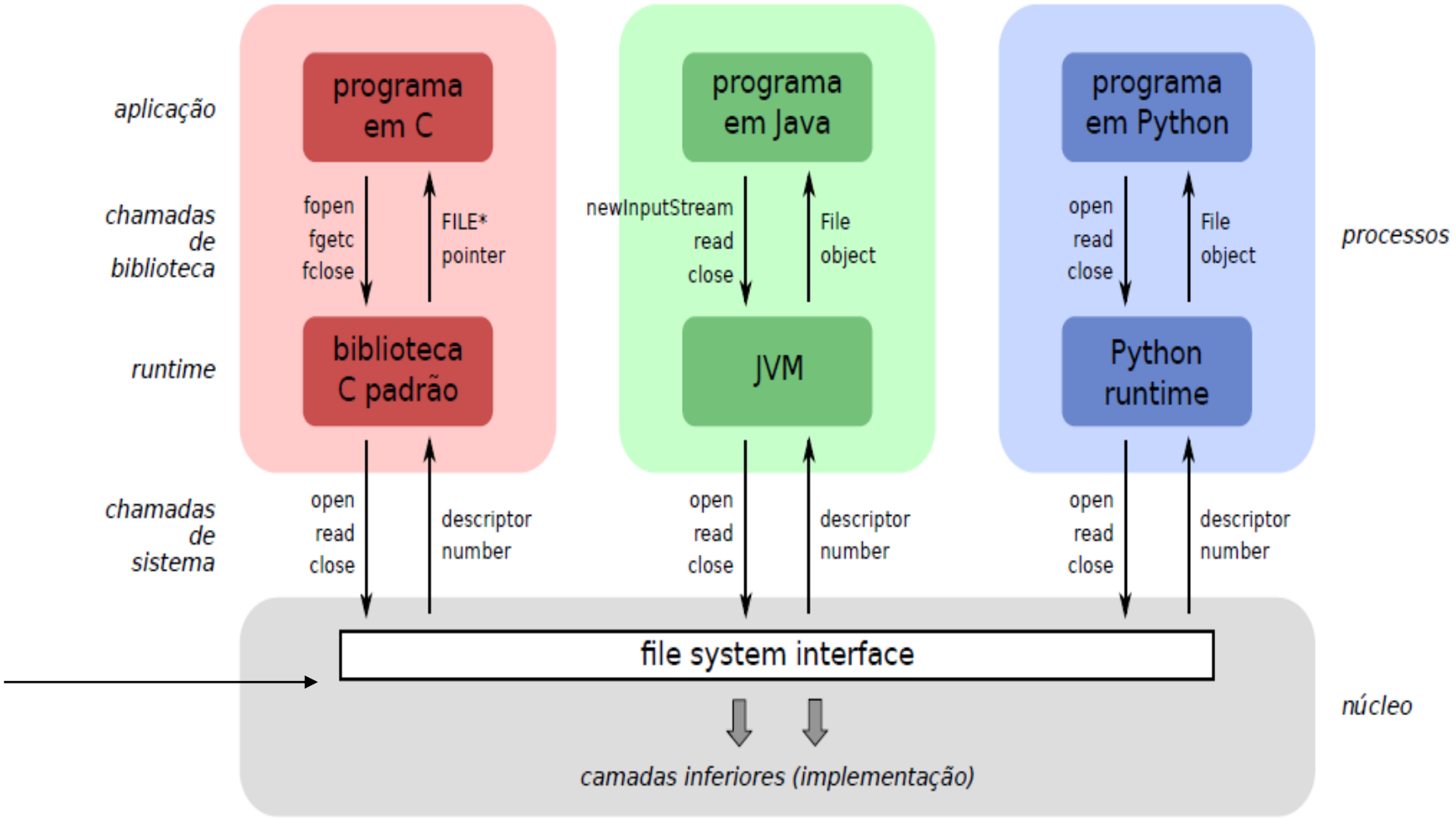
Sistemas de Arquivos

- 3 critérios para armazenamento de informações persistentes:
  - Deve ser capaz de armazenar uma quantidade **muito grande** de informações
  - A informação deve **sobreviver** aos processos que a utilizam
  - Deve fornecer acesso **simultâneo** a vários processos

- Solução:
  - Armazenar informações em discos em unidades chamadas de arquivos
  - Os arquivos são persistentes e somente o proprietário pode excluí-los explicitamente
  - Os arquivos são gerenciados pelo sistema operacional

- ***Sistemas de arquivos***: como o sistema operacional gerencia arquivos!

Sistemas de Arquivos



Atributos de Arquivos

- Informações específicas do arquivo mantidas pelo sistema operacional
  - Tamanho do arquivo, data de modificação, hora de criação, etc.
- Varia muito em diferentes sistemas operacionais
- Alguns exemplos:
  - Nome – apenas informações mantidas em formato legível por humanos
  - Identificador – tag exclusiva (número) identifica o arquivo dentro do sistema de arquivos
  - Tipo – necessário para sistemas que suportam diferentes tipos
  - Localização – ponteiro para a localização do arquivo no dispositivo
  - Tamanho – tamanho do arquivo atual
  - Proteção – controla quem pode ler, escrever, executar
  - Hora, data e identificação do usuário – dados para proteção, segurança e monitoramento de uso

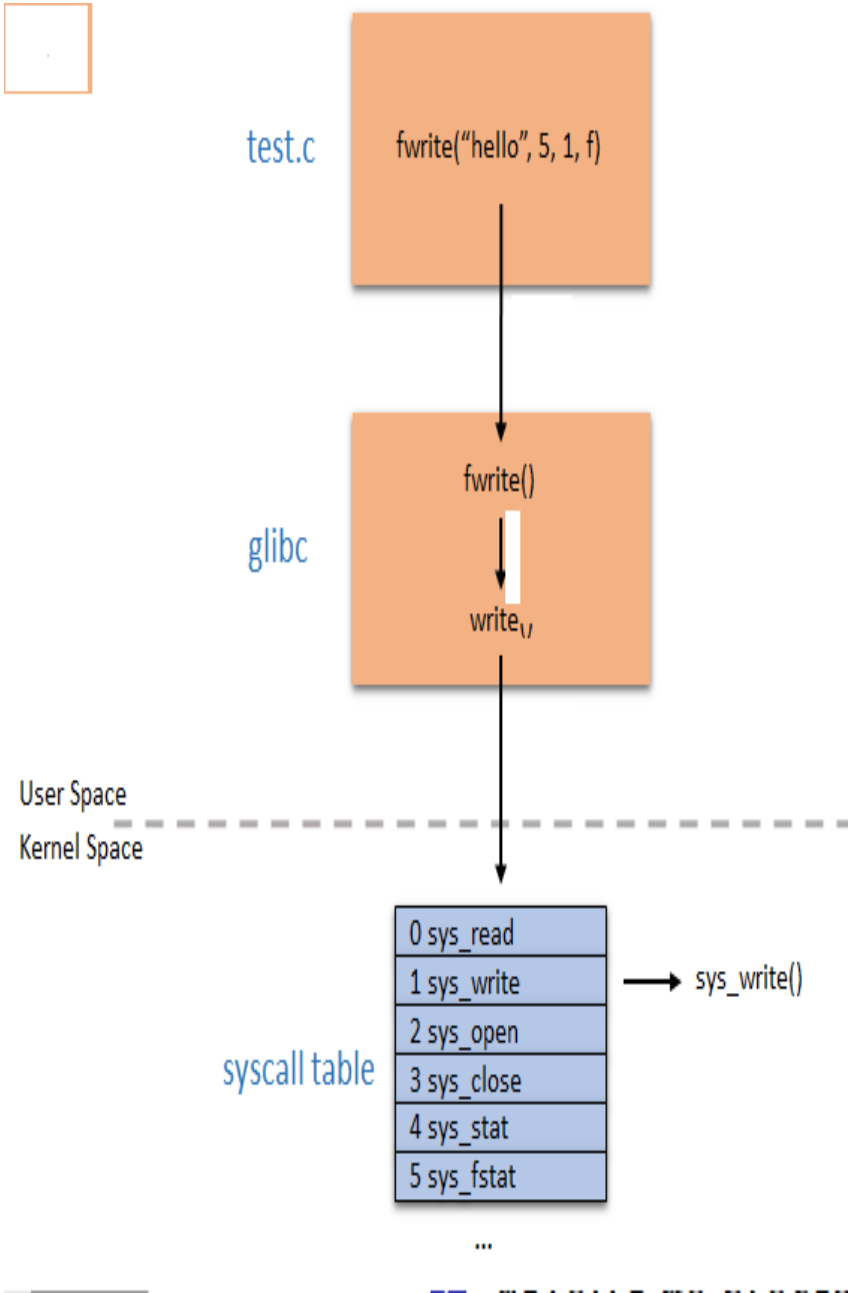
Operações Básicas

Operação	C (padrão C99)	Java (classe File)
Abrir arquivo	`fd = fopen(...)`	`obj = File(...)`
Ler dados	`fread(fd, ...)`	`obj.read()`
Escrever dados	`fwrite(fd, ...)`	`obj.write()`
Fechar arquivo	`fclose(fd)`	`obj.close()`
Remover arquivo	`remove(...)`	`obj.delete()`
Criar diretório	`mkdir(...)`	`obj.mkdir()`

Operação	Linux	Windows
Abrir arquivo	OPEN	NtOpenFile
Ler dados	READ	NtReadRequestData
Escrever dados	WRITE	NtWriteRequestData
Fechar arquivo	CLOSE	NtClose
Remover arquivo	UNLINK	NtDeleteFile
Criar diretório	MKDIR	NtCreateDirectoryObject

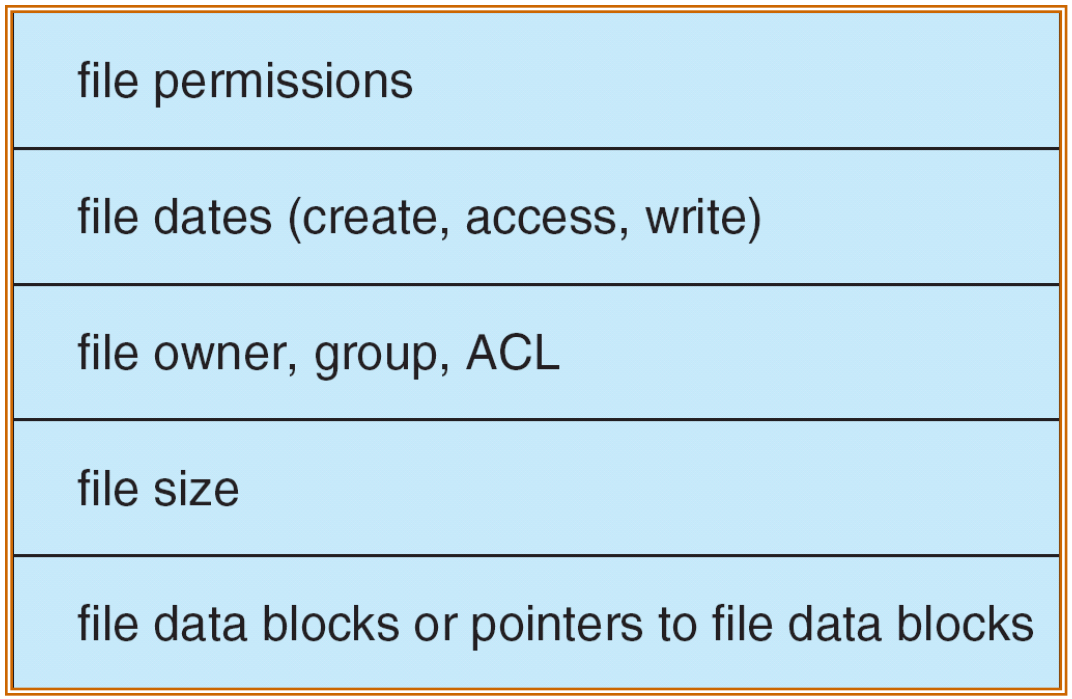
Exemplo

- No processo:
  1. a aplicação pede a abertura do arquivo (`fopen()`)
  2. o *runtime* da linguagem invoca uma *syscall*
- No núcleo:
  3. o núcleo localiza o arquivo no dispositivo físico
  4. verifica as permissões de acesso do arquivo
  5. cria um descritor para o arquivo aberto no núcleo
  6. insere o descritor em uma lista de arquivos abertos
  7. devolve ao processo uma referência ao descritor
- No processo:
  8. o *runtime* recebe o descritor de baixo nível
  9. o *runtime* cria um descritor de alto nível
  10. a aplicação recebe o descritor de alto nível

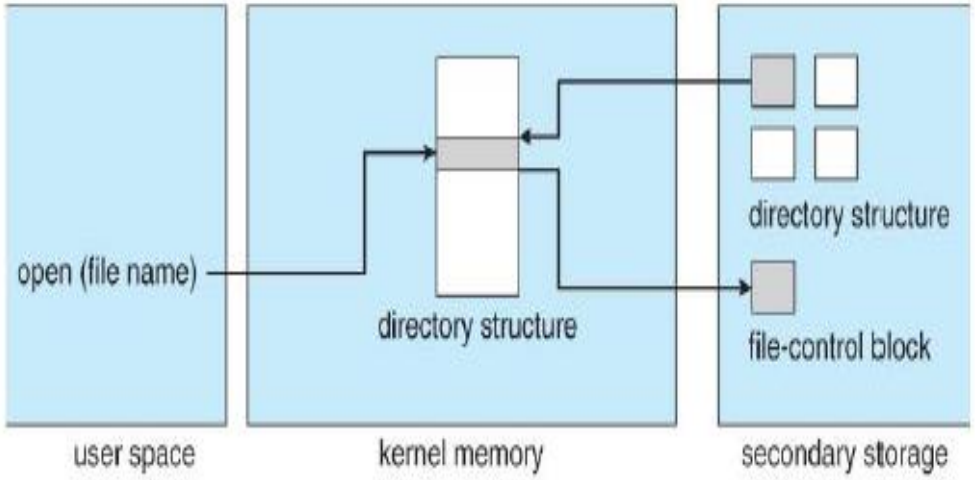


File Control Block (FCB)

- FCB tem todas informações do arquivo

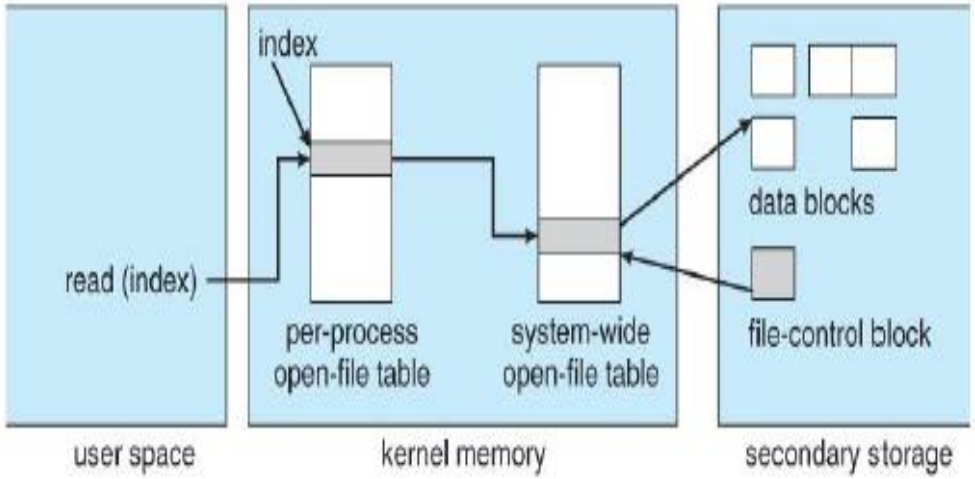


File Control Block (FCB)



FCB: File Control Block

Contém **metadados** (tamanho, permissões, data, etc) sobre o arquivo



file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

Controle de acesso

Definir e controlar que usuários podem acessar que arquivos.

Atributos relevantes:

- **Proprietário**: usuário dono do arquivo
- **Permissões**: operações permitidas aos usuários

Implementação:

- Definição de usuários e grupos
- Listas de controle de acesso (ACL)
- O núcleo autoriza ou nega cada acesso

ACL

Access Control List associada a cada arquivo.

Indica as ações que os usuários podem fazer sobre ele.

Exemplo:

```
1 arq1.txt  : (João: ler), (José: ler, escrever), (Maria: ler, remover)
2 video.avi : (José: ler), (Maria: ler)
3 musica.mp3: (Daniel: ler, escrever, apagar)
```

Permissões em UNIX

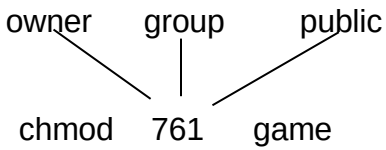
- Usuários:
  - `user` : o proprietário do arquivo
  - `group`: um grupo de usuários associado ao arquivo
  - `other`: os demais usuários
- Permissões: `read`, `write`, `execute`

Exemplo Linux

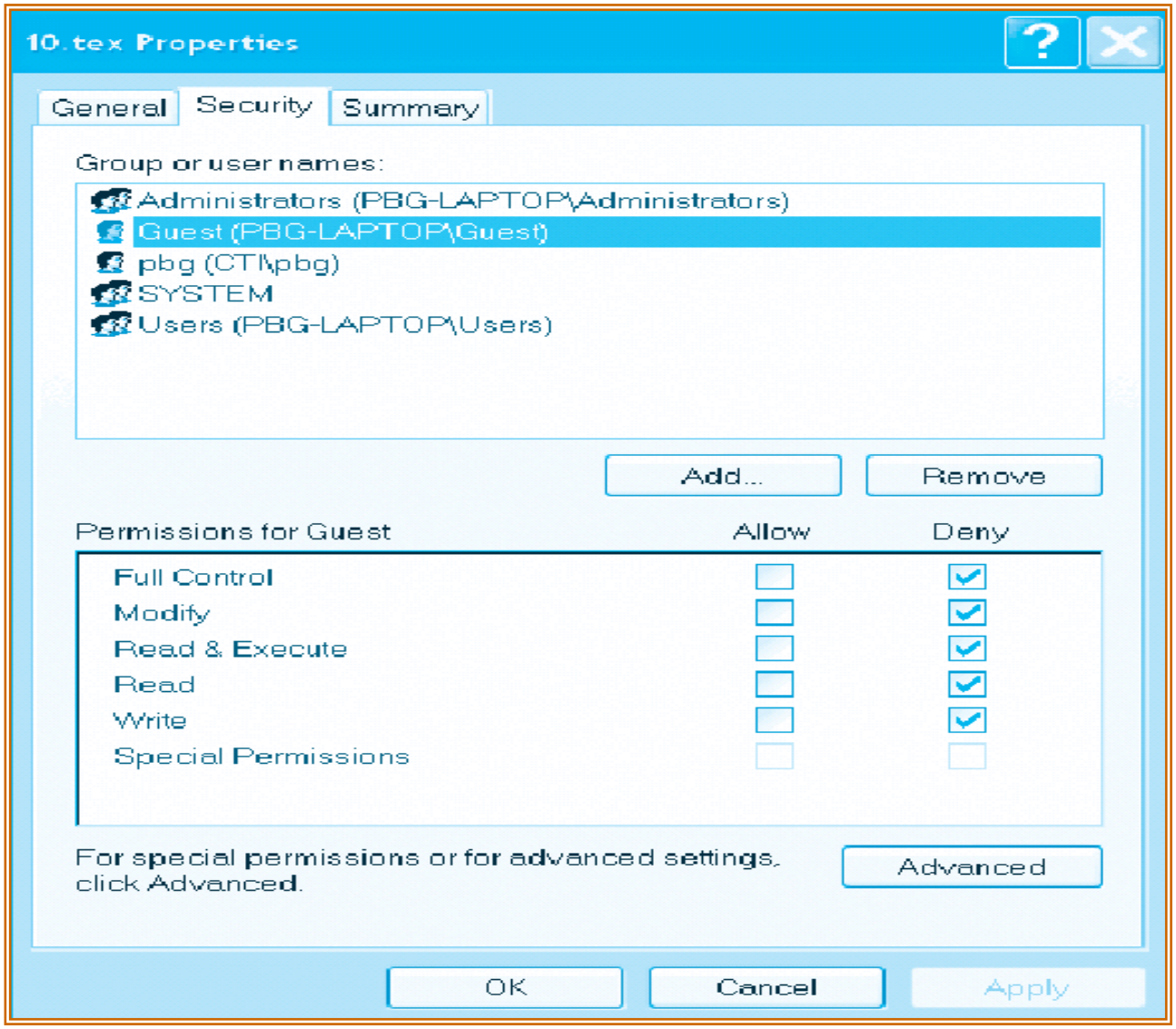
- Modos de acesso: read, write, execute

a) **owner**	7	⇒	1 1 1 RWX
b) **group**	6	⇒	1 1 0 RWX
c) **public**	1	⇒	0 0 1

- Exemplo:



Exemplo: windows



Exemplos de Sistemas de Arquivos

- FAT-16: Windows 95 e Windows 98
- FAT-32: Introduzido pelo Windows 98
- exFAT: Versão otimizada do FAT para flash drivers e grandes sistemas de arquivos.
- NTFS: Introduzido pelo Windows NT e suportado até hoje
- ReFS: Introduzido na versão Server do Windows 8
- ext/ext2: Usados nas primeiras versões do Linux (até 2001)
- ext3: mais confiável quanto a corrupção de arquivos
- ext4: melhoria introduzida em 2006 (suporte a discos maiores)

Sistema de Arquivos: Windows

Features	NTFS	FAT32	FAT16	FAT12
Max Partition Size	2TB	32GB	4GB	16MB
Max File Size	16TB	4GB	2GB	Less than 16MB
Cluster Size	4KB	4KB to 32KB	2KB to 64KB	0.5KB to 4KB
Fault Tolerance	Auto Repair	No	No	No
Compression	Yes	No	No	No
Security	Local and Network	Only Network	Only Network	Only Network
Compatibility	Windows 10/8/7/XP/Vista/2000	Windows ME/2000/XP/7/8.1	Windows ME/2000/XP/7/8.1	Windows ME/2000/XP/7/8.1

Sistema de Arquivos Linux

Point	ext2	ext3	ext4
Maximum individual file size	16GB – 2TB	16GB – 2TB	16GB – 16TB
Maximum file system size	2TB – 32TB	2TB – 32TB	1EB
Journalling	Not available	Available	Available and can be turned "off" too
Number of directories	31998	31998	Unlimited
Journal checksum	No	No	Yes

Tamanho de bloco default: 4K

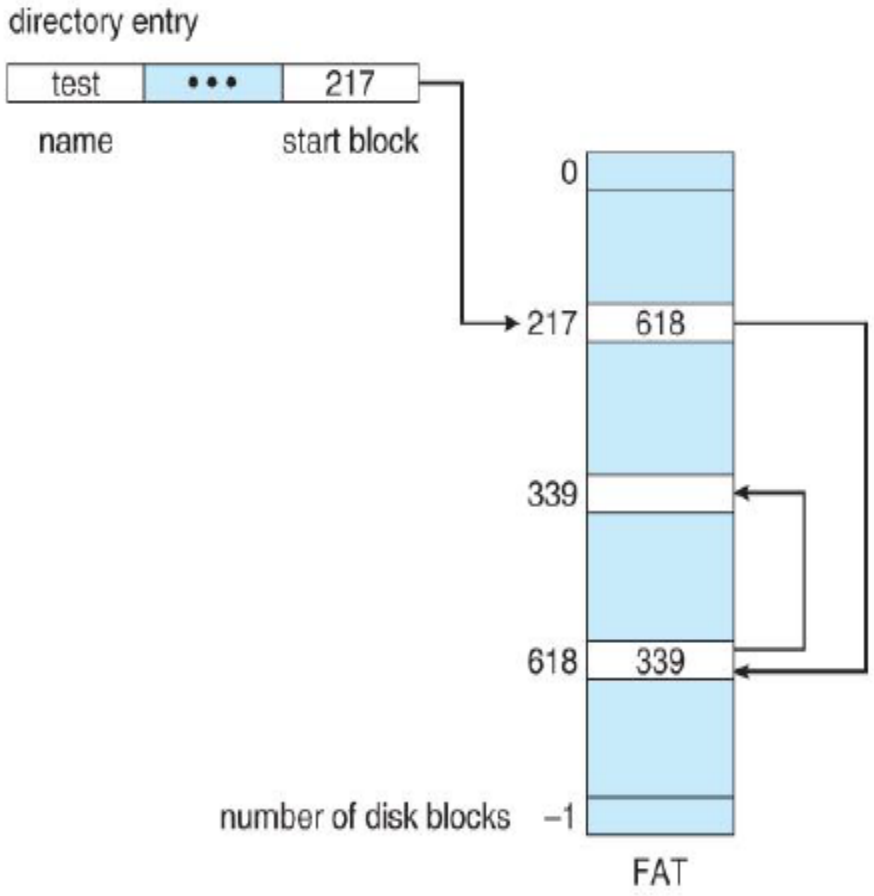
Journaling é um mecanismo que aumenta a confiabilidade contra corrupção de arquivos em caso de crash, salvando as alterações que ainda não foram escritas em disco em outro local (journal).

1 Byte = 8 bits.
1 Kilobyte = 1024 bytes
1 Megabyte = 1024 Kilobytes ou KB
1 Gigabyte = 1024 Megabytes ou MB
1 Terabyte = 1024 Gigabytes ou GB
1 Petabyte = 1024 Terabytes ou TB
1 Exabyte = 1024 Petabytes ou PB
1 Zettabyte = 1024 Exabytes ou EB
1 Yottabyte = 1024 Zettabytes ou ZB

FAT (File Allocation Table)

mapeamento do arquivo em blocos

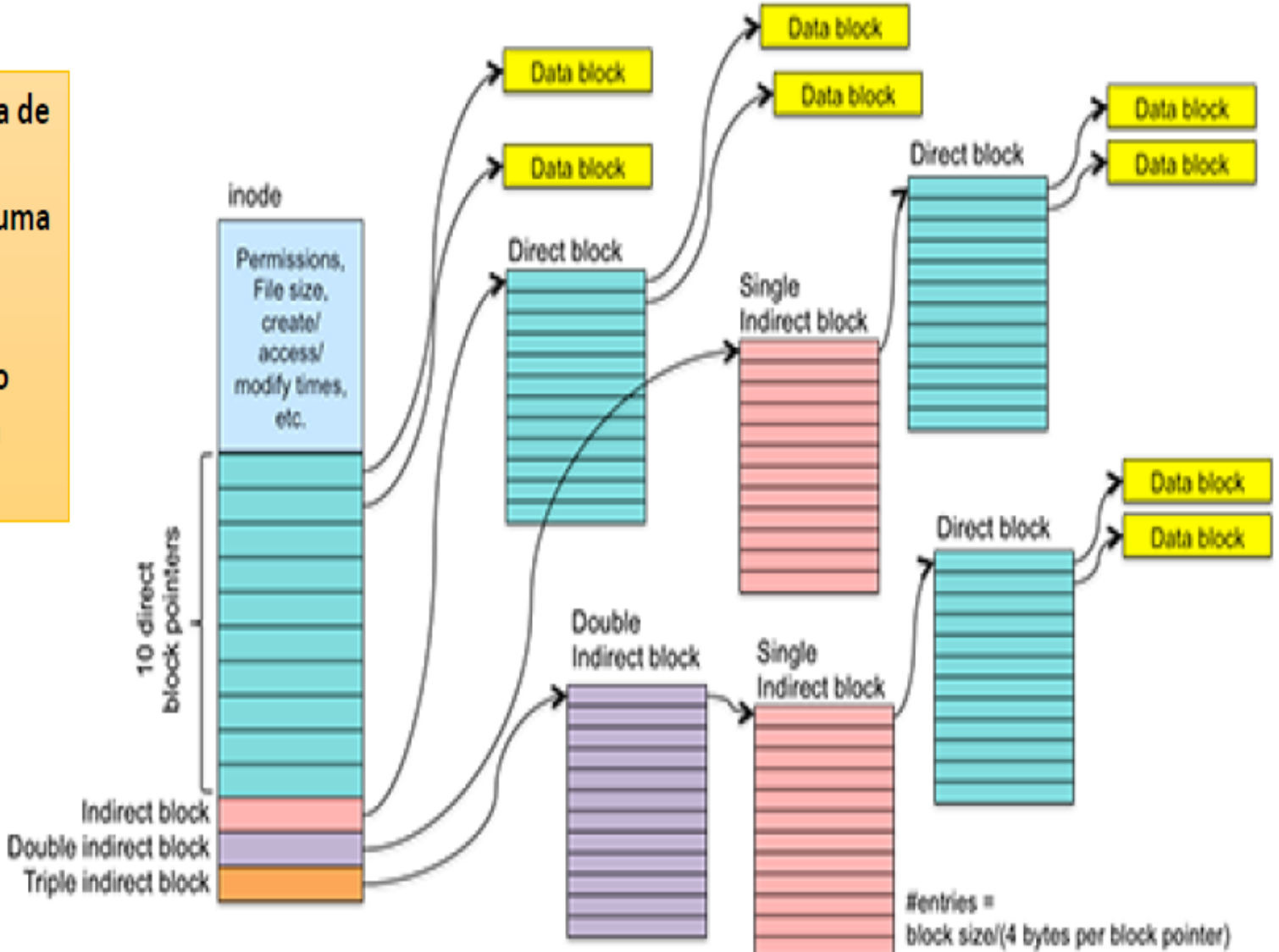
- FAT é uma tabela armazenada no início de cada partição do disco.
- As entradas da tabela correspondem aos blocos da partição do disco.
- Cada entrada aponta para o próximo bloco usado pelo arquivo.



iNodes

mapeamento do arquivo em blocos

Cada entrada de diretório é associada a uma estrutura de dados de tamanho fixo denominada inode



Exercício

- Ative uma máquina virtual LINUX disponível no notebook
- 1) Comando: $ sudo fdisk –l ; sudo fdisk –l /dev/sda
  - Qual o tamanho do setor de disco?
  - Quais os discos mostrados no comando? Qual tamanho?
  - Existem partições no disco?
- 2) Comando: $ sudo file -Ls /dev/sda1
  - Qual o tipo de sistema de arquivos do seu disco?
- 3) Comando: $ sudo blkid /dev/sda1
  - Qual Id da partição?
- 4) Comando: $ sudo df -h
  - Quais os tamanhos e espaços disponíveis?
- 5) Crie um arquivo e vamos ver informações de inode dele:
  - $ nano teste.txt
  - $ stat teste.txt
  - Analise as informações do arquivo.
- 6) Consulte o arquivo de configuração das partições a serem montadas no boot
  - # cat /etc/fstab
  - Analise as informações

- **File**: The name of the file. Usually, it is the same as the name we passed to `stat` on the command line, but it can be different if we're looking at a symbolic link.
- **Size**: The size of the file in bytes.
- **Blocks**: The number of filesystem blocks the file requires, in order to be stored on the hard drive.
- **IO Block**: The size of a filesystem block.
- **File type**: The type of object the metadata describes. The most common types are files and directories, but they can also be links, sockets, or named pipes.
- **Device**: The device number in hexadecimal and decimal. This is the ID of the hard drive the file is stored on.
- **Inode**: The inode number. That is, the ID number of this inode. Together, the inode number and the device number uniquely identify a file.