

# Sistema de arquivos

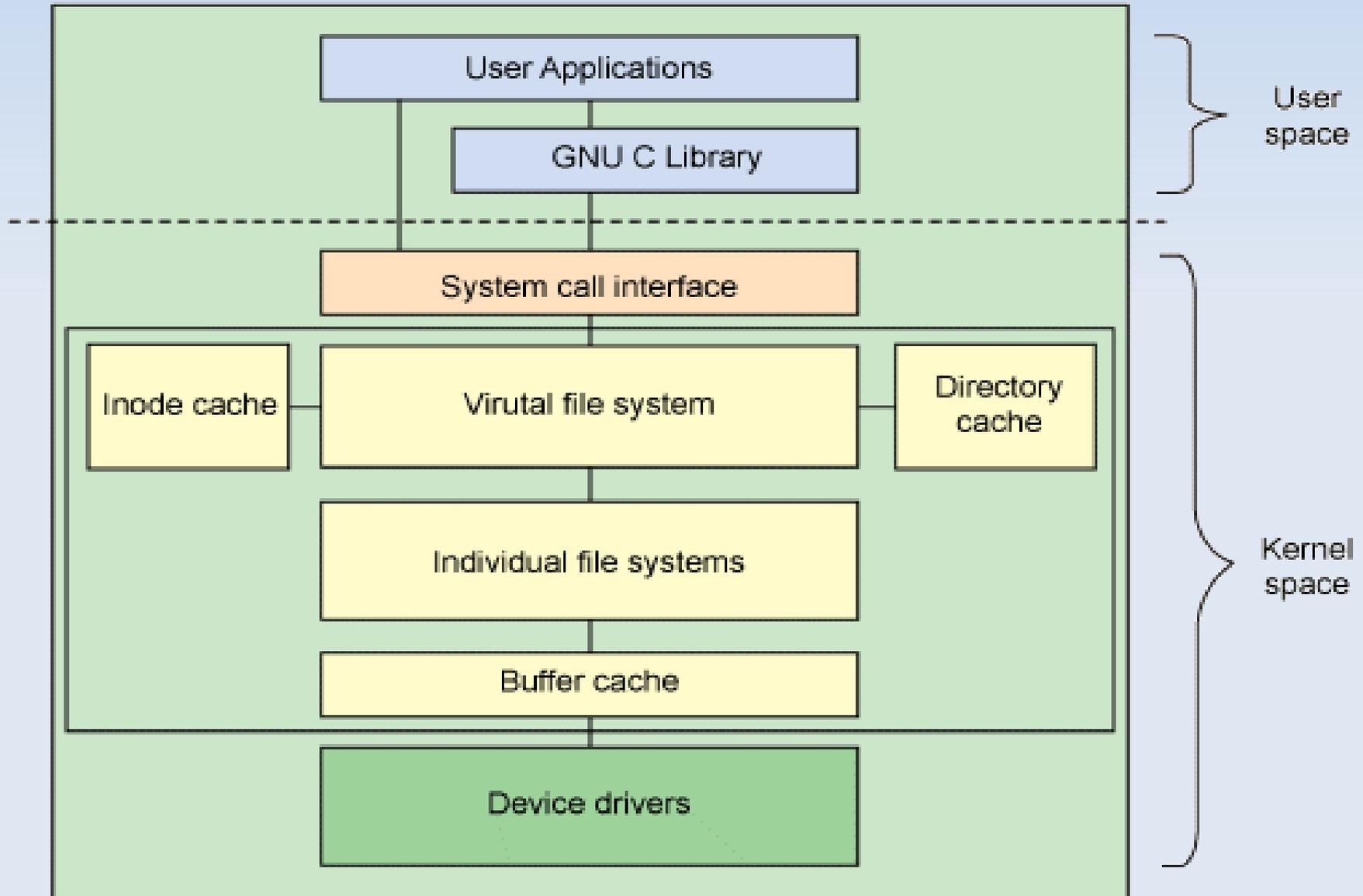
## Tópicos

- VFS
- Classificações de arquivos
- Registro inicial dos sistemas de arquivos
- Propriedades e atributos
- Ferramentas e manipulação de arquivos

# Sistema de arquivos em Linux

- API implementada pelo VFS
- Características dos sistemas UNIX
- Árvore de diretórios padrão FHS
- Sistema de arquivos unificado (diversos dispositivos em um mesma árvores de diretórios com base no *root filesystem*)
- Pontos de montagem (diretórios) sobre as partições (dispositivos)

# Sistema de arquivos em Linux



# VFS

- Estruturas suportadas pelo VFS: dentry, file, inode e superblock
- Superblock operations
- Inode, dentry e file operations
- Função do VFS:
  - Manter controle dos FS disponíveis
  - Associar e desassociar dispositivos com instancias do FS
  - Processar as operações dos objetos. Para procedimentos genéricos, realizar em buffer. Para específicos, acionar diretamente a camada do FS

# VFS

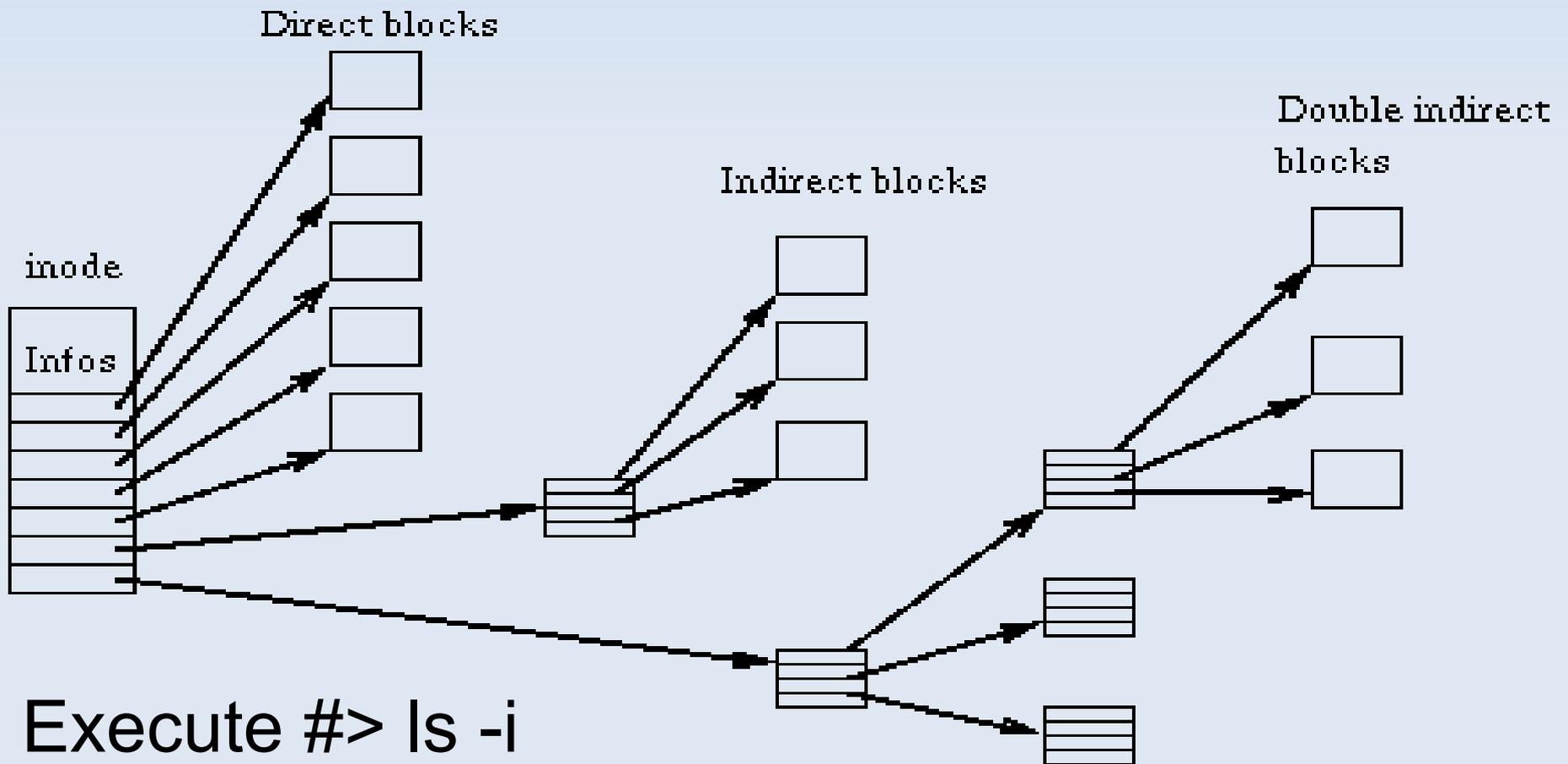
- VFS torna-se uma camada entre o FS e a chamada de sistema
- Todo FS a ser utilizado no sistema deve se registrar no VFS através de operações
- Através de módulos ou procedimentos de kernel é possível interagir com diversos tipos de sistemas de arquivos: ext, xfs, jfs, fat32, ntfs, iso9660, reiserfs, etc

# Sistemas de arquivos mais utilizados

- ReiserFS e ext4
- Utilização de técnica de "journal filesystem"
- Organização de estrutura de diretórios através de árvores (b-tree)

# Estruturas

- Estrutura de Inode em sistema EXT



# Tipos de arquivos em Linux

- Regulares – binários ou ASCII
- Diretórios
- Links – simbólicos ou de hardware
- Dispositivos
- Pipes
- Sockets

# Atividade

Crie ao menos cada um dos tipos de arquivos através dos comandos

- touch
- mkdir
- mkfifo
- mknod

# Inicialização do FS

- Organização do arquivo /etc/fstab

<file system> <mount point> <type> <options><dump><pass>

# Inicialização do FS

file system - dispositivo com partição (identificação do FS)

mount point – diretório ou entrada na árvore

type - tipo do sistema de arquivos (ext4, reiserfs, etc)

options – permissões e atributos (ro,rw,user,etc)

dump – opção para ferramenta de backup 1 ou 0

pass – opção para ligar ou desligar a verificação de consistência

# Ferramentas e utilização

Exemplos:

```
#> mount
```

Montagem:

```
#> mount -t msdos /dev/sdb1 /mnt/pendrive
```

```
#> mount -t nfs remoto:/tmp /mnt/computador_remoto_temp
```

Desmontagem:

```
#> umount /mnt/pendrive
```

```
#> umount /dev/sdb1
```

# Atividade

- Faça a montagem de um pendrive em um diretório fora do padrão automatizado pelo sistema (mount e umount), por exemplo, utilizando a pasta /pendrive.

# Automatizando montagem

- autofs dameon
- hal dameon
- manter uma entrada no registro no /etc/fstab
- utilização do udevd para auxílio

# Atividade

Criando uma partição e montando "online"

```
#> dd if=/dev/zero of=file.img bs=1k count=10000
```

```
#> losetup /dev/loop0 file.img
```

```
#> mke2fs -c /dev/loop0 10000
```

```
#> mkdir /mnt/point1
```

```
#> mount -t ext2 /dev/loop0 /mnt/point1
```

```
#> ls /mnt/point1
```

# Ferramentas e utilização

Principais ferramentas para criação e manutenção de partições

- fdisk

- mkfs

- hdparm

- fsck

# Atividade

- Através dos comandos **df** e **du** descubra os espaços disponíveis nas partições já montadas ou em diretórios específicos em seu sistema de arquivos:

# Controle e atributos dos arquivos

© Original Artist  
Reproduction rights obtainable from  
[www.CartoonStock.com](http://www.CartoonStock.com)



search ID: aba0345

**"According to Einstein's theory, if we move the computer real fast, we can go back in time and recover the files you accidentally deleted."**

# Atributos dos arquivos

```
drwx----- 3 zuolo zuolo 4096 2010-09-21 18:00 .kde
drwxr-xr-x 2 zuolo zuolo 4096 2010-09-21 18:01 .xine
-rw-r--r-- 1 root root 11 2010-09-21 18:03 compra
-rw-r--r-- 1 zuolo zuolo 75675 2010-09-22 12:09 brasao_assinatura_email.jpg
-rw-r--r-- 1 zuolo zuolo 1181 2010-09-22 12:11 assinatura.html
-rw-r--r-- 1 zuolo zuolo 201 2010-09-22 12:14 .signature
drwxr-xr-x 5 zuolo zuolo 4096 2010-09-22 13:39 .gkrellm2
-rw----- 1 zuolo zuolo 40 2010-09-22 17:13 .lesshst
-rw-r----- 1 zuolo zuolo 0 2010-09-22 18:48 .gksu.lock
drwxr-xr-x 2 zuolo zuolo 4096 2010-09-22 18:50 .themes
drwxr-xr-x 2 zuolo zuolo 4096 2010-09-22 18:50 .icons
drwx----- 4 zuolo zuolo 4096 2010-09-22 18:50 .thumbnails
-rw----- 1 zuolo zuolo 2142 2010-09-22 18:51 .ICEauthority
drwx----- 2 zuolo zuolo 4096 2010-09-22 18:51 .pulse
```

# Atributos dos arquivos

```
drwx----- 3 zuolo zuolo 4096 2010-09-21 18:00 .kde
drwxr-xr-x 2 zuolo zuolo 4096 2010-09-21 18:01 .xine
-rw-r--r-- 1 root root 11 2010-09-21 18:03 compra
-rw-r--r-- 1 zuolo zuolo 75675 2010-09-22 12:09 brasao_assinatura_email.jpg
-rw-r--r-- 1 zuolo zuolo 1181 2010-09-22 12:11 assinatura.html
-rw-r--r-- 1 zuolo zuolo 201 2010-09-22 12:14 .signature
drwxr-xr-x 5 zuolo zuolo 4096 2010-09-22 13:39 .gkrellm2
-rw----- 1 zuolo zuolo 40 2010-09-22 17:13 .lesshst
-rw-r----- 1 zuolo zuolo 0 2010-09-22 18:48 .gksu.lock
drwxr-xr-x 2 zuolo zuolo 4096 2010-09-22 18:50 .themes
drwxr-xr-x 2 zuolo zuolo 4096 2010-09-22 18:50 .icons
drwx----- 4 zuolo zuolo 4096 2010-09-22 18:50 .thumbnails
-rw----- 1 zuolo zuolo 2142 2010-09-22 18:51 .ICEauthority
drwx----- 2 zuolo zuolo 4096 2010-09-22 18:51 .pulse
```

# Alterando permissões

### chmod

Utiliza 3 classes de usuários, cada uma com 3 bits de acesso (read, write e execute) rwx (7 decimal), rw- ( 6 decimal), r-x(5 decimal), r-- ( 4decimal), -wr (3 decimal) ... etc

Exemplos

```
#> chmod 765 /var/log/messages
```

```
7 ---- usuário proprietário  rwx (421)
```

```
6 ---- grupo proprietário    rw- (42-)
```

```
5 ---- outros                 r-x (4-1)
```

```
#> umask (altera máscara padrão)
```

# Alterando permissões

### Além de utilizar a notação decimal-binária, pode-se utilizar notações para cada classe e permissão como: u=user, g=group, o=other, a=all, +(adiciona), -(remove), r(leitura),w (escrita) e x(execução).



# Atividade

Crie um arquivo vazio qualquer e altere suas permissões, fazendo com que:

- usuário tenha direito de escrita, leitura; grupo tenha somente direito de execução e outros tenham direito de execução e escrita.

Faça isso com as duas sintaxes diferentes do comando *chmod*

O que faz o comando *stat* ?

# Alterando proprietários

### chown

Exemplos

```
#> chown dono /var/log/messages
```

```
#> chown .grupo /var/log/messages
```

```
#> chown dono.grupo /var/log/messages
```

```
#> chown -R dono.grupo /var/log/
```

# Atributos especiais

### suid sgid stick rwx (user) rwx (group) rwx (other)

suid – bit que força a execução do arquivo através do proprietário do mesmo

sgid - bit que força a execução do arquivo através do grupo do mesmo

stick – congela os atributos do arquivos, permitindo somente ao proprietário remover ou renomear tais arquivos

Utiliza o mesmo conceito dos atributos de permissão, mas adiciona mais 3 bits iniciais aos 9 já existentes para usuários

```
#> chmod 7754 /var/log/messages
```

Ativa suid, sgid e stick, além de permitir rwx para usuário, rx para grupo e r para outros.

# Atividade

- Crie um arquivo dentro do diretório /tmp e faça com que o mesmo tenha `suid` e `stickbit` ligados