



A partição EXT3 (*Third Extended Filesystem*) é usada para criar o sistema de arquivos Linux Native que serve para armazenar o sistema de arquivos EXT3 (após a formatação) e permitir o armazenamento de dados.

Logo que foi inventado, o Linux utilizava o sistema de arquivos Minix (e consequentemente uma partição Minix) para o armazenamento de arquivos. Com a evolução, foi criado o padrão EXT (*Extended Filesystem*) e logo o EXT2 (*Second Extended Filesystem*), que ainda é usado atualmente.

O sistema EXT3 implementa o conceito de *journaling*. O sistema de *journaling* grava qualquer operação que seja feita no disco em uma área especial chamada *journal*, assim, se acontecer algum problema durante a operação de disco, ele pode voltar ao estado anterior do arquivo, ou finalizar a operação.

Desta forma, o *journal* acrescenta ao sistema de arquivos o suporte da alta disponibilidade e da maior tolerância a falhas. Após uma falta de energia, por exemplo, o *journal* é analisado durante a montagem do sistema de arquivos e todas as operações que estavam sendo feitas no disco são verificadas. Dependendo do estado das operações, elas podem ser desfeitas ou finalizadas. O retorno do servidor é praticamente imediato, garantindo a rápida retomada dos serviços da máquina.

Outra situação que pode ser evitada é a inconsistência no sistema de arquivos do servidor após a situação acima, fazendo-o ficar em estado *single user* (usuário único), esperando pela intervenção do administrador.

## Arquivos

Os arquivos são gerenciados pelo sistema operacional e é mediante a implementação deles que o sistema operacional estrutura e organiza as informações. Um arquivo é constituído de informações logicamente relacionadas, podendo representar programas ou dados.

Os arquivos são identificados por meio de um nome formado por uma sequência de caracteres. A identificação de um arquivo é composta por duas partes separadas por um ponto. A parte após o ponto é chamada extensão do arquivo e “serve” para identificar o conteúdo. Exemplos:

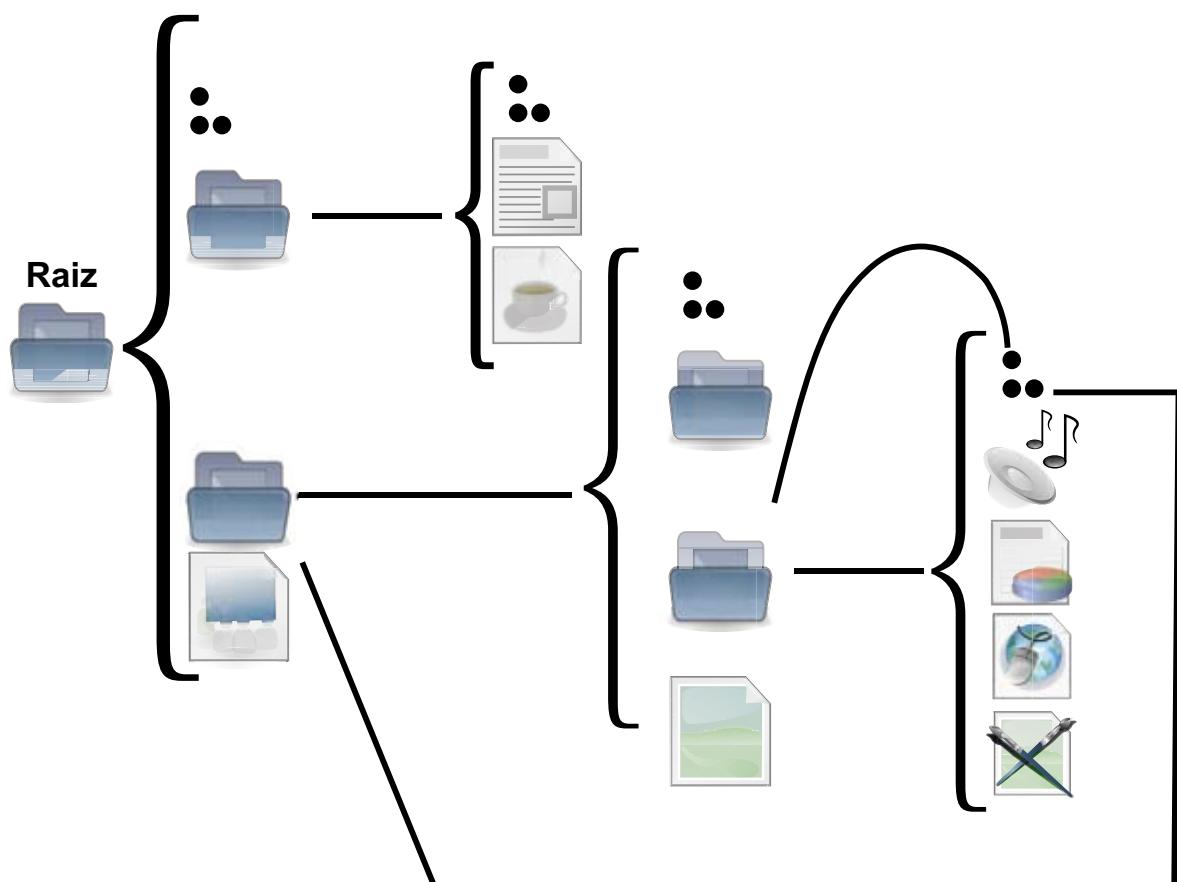
- **arquivo.c** – Arquivo fonte em C;
- **arquivo.mp3** – Arquivo de música;
- **arquivo.dll** – Biblioteca dinâmica.

# Diretórios

É o modo como o sistema organiza os diferentes arquivos contidos em um disco. Utilizando um método relativamente simples, o diretório é implementado como um arquivo estruturado, cujo conteúdo é uma relação de entradas na forma de uma estrutura de dados. Cada estrutura contém entradas associadas aos arquivos onde estão informações, como localização física, nome, organização e demais atributos.

Os tipos de entradas normalmente consideradas nessa relação são arquivos normais, diretórios, atalhos e entradas associadas a arquivos especiais. Cada entrada contém ao menos o nome do arquivo (ou do diretório), seu tipo e a localização física dele na partição.

O Linux utiliza algumas entradas padronizadas. A entrada “.” (ponto), que representa o próprio diretório, e a entrada “..” (ponto-ponto), que representa seu diretório pai (o diretório imediatamente acima dele na hierarquia de diretórios). No caso do diretório raiz, ambas as entradas apontam para ele próprio. Veja a figura a seguir.



## Diretórios Especiais “.” e “..”

Toda vez que um diretório é criado, sempre são criadas duas entradas nele. Uma entrada, com o nome de “.”, referencia-se ao próprio diretório criado e a outra entrada, com o nome de “..”, uma referência ao diretório anterior, ou diretório pai, na estrutura do sistema de arquivos.

Estes dois arquivos podem ser usados para compor qualquer caminho, relativo ou absoluto, dentro dos comandos do Linux, e visam a facilitar a digitação de comandos.

Normalmente, eles não aparecem na relação de arquivos, pois o sistema esconde todos os arquivos que começam com um ponto em seu nome. Para que se listem estas duas entradas, devemos usar a opção **-a** do comando **ls**.

## Caminho de Arquivos

O sistema de arquivos é apresentado como uma única hierarquia unificada que se inicia no diretório **/** e prossegue abaixo até um número arbitrário de subdiretórios. O diretório **/** também é chamado de diretório-raiz.

Sempre que precisamos localizar um arquivo para qualquer operação (ler, gravar, remover, criar, etc.) o sistema operacional deve conhecer em que ponto do sistema de arquivos ele se encontra. Isto é feito por meio da especificação de um caminho antes do nome do arquivo. Esse caminho, chamado de **path**, pode ser indicado de duas maneiras:

- **Absoluto** – o caminho absoluto sempre começa com uma barra **/**. Esse caminho dá a localização do arquivo desde o diretório-raiz do sistema. O sistema operacional começa pela raiz e vai seguindo os diretórios indicados até o último.
- **Relativo** – a procura de um arquivo por meio de um caminho relativo começa no próprio diretório atual da sessão.

## Comandos Básicos para Trabalhar com Diretórios

Os comandos usados para navegação na árvore de diretórios são similares aos usados em outros sistemas operacionais:

- **pwd**: indica qual o diretório corrente do shell.
- **cd**: troca de diretório.
- **cd dir**: muda para o diretório dir.
- **cd ..**: muda para o diretório pai imediatamente superior.
- **cd -**: volta para o último diretório visitado.
- **cd**: volta ao diretório HOME.

Imprime o nome do diretório corrente.

```
$ pwd  
/home/marcos
```

Acessa o diretório /etc (utilizando caminho absoluto, ou seja, utiliza o /).

```
$ cd /etc
```

Imprime o nome do diretório corrente.

```
$ pwd  
/etc
```

Acessa o diretório apt e imprime o nome do diretório corrente.

```
cd apt  
$ pwd  
/etc/apt
```

Acessa o diretório home (comando **cd** sem argumentos) e imprime o nome do diretório corrente.

```
$ cd  
$ pwd  
/home/marcos
```

Acessa o diretório anterior (comando **cd** com argumento -).

```
$ cd -  
/etc/apt
```

Acessa o diretório anterior utilizando caminho relativo.

```
$ cd ..  
$ pwd  
/home
```

Acessa o diretório/etc (utilizando caminho relativo, ou seja, informando o caminho a partir da posição atual).

```
$ cd ../etc  
$ pwd  
/etc
```

## Caminho relativo ou absoluto, qual a melhor opção?

Considere a árvore de diretório abaixo.

```
\  
|-- bin  
|-- boot  
|  |-- boot  
|  |  `-- grub  
|  '-- grub  
|-- etc  
|  |-- X11  
|  |-- acpi  
|  |-- alternatives  
|  '-- apt  
|-- home  
|  '-- marcos  
|    |-- Documentos  
|    |-- Downloads  
|    |-- Imagens  
|    |-- Modelos  
|    '-- uml  
|-- lib  
|-- mnt  
`-- opt
```

Suponha que você está posicionado no diretório **Documentos** e deseja acessar o diretório **Downloads**. Utilizando caminho relativo, você usaria o seguinte comando:

**cd .../Downloads** (indica que você quer descer um nível e subir para o diretório Downloads.)

Utilizando o caminho absoluto (ou seja, informar o caminho completo a partir do raiz).

**cd /home/marcos/Downloads**

Ambos os comandos fazem a mesma coisa, só que no primeiro caso você digitou menos palavras. Agora considere que você está no diretório **Downloads** e deseja acessar o diretório **etc**. Utilizando caminho relativo:

**cd ../../etc** (desce três níveis para depois subir para o diretório **etc**).

Utilizando o caminho absoluto (ou seja, informar o caminho completo a partir do raiz):

**cd /etc**

Fica evidente que, neste caso, é mais vantajoso utilizar o caminho absoluto.

Ao trabalhar com arquivos e diretórios, o uso dos caminhos absolutos ou relativos fica a critério de cada usuário.

- **mkdir dir**: criação do diretório dir.
- **rmdir dir**: remoção do diretório dir.

Cria o diretório teste\_aula e acessa o diretório teste\_aula.

```
$ mkdir teste_aula  
$ cd teste_aula  
$ pwd  
/home/marcos/teste_aula
```

Volta ao diretório anterior e apaga o diretório.

```
$ cd ..  
$ rmdir teste_aula
```

Tenta acessar o diretório que foi apagado anteriormente.

```
$ cd teste_aula  
bash: cd: teste_aula: Arquivo ou diretório não encontrado
```

## Comandos Básicos para Trabalhar com Arquivos

Os comandos a seguir implementam operações básicas em arquivos:

- **ls** – listar o conteúdo do diretório corrente (ou de um diretório dado).

```
$ ls -l /etc  
total 1488  
-rw-r--r-- 1 root      root        149 2009-07-13 22:25 00-header  
drwxr-xr-x  4 root      root      4096 2009-10-27 16:10 acpi  
-rw-r--r--  1 root      root     2986 2009-10-27 15:57 adduser.conf  
drwxr-xr-x  2 root      root      4096 2009-11-23 20:54 alternatives  
-rw-r--r--  1 root      root      395 2009-09-17 16:33 anacrontab  
  
<----- resultado suprimido propositalmente ----->
```

- **rm** – este comando serve para eliminar um arquivo do sistema de arquivos. Ele também fará a remoção de diretórios se for especificada a opção **-r**. Com essa opção todo o diretório é excluído, inclusive todos os seus subdiretórios, indiferente se ele possui ou não arquivos. A opção **-f** (**método forçado**) indica que nenhuma confirmação deve ser solicitada ao usuário, ou seja, o comando deverá excluir arquivos e diretórios que estejam com permissão apenas de leitura ou sem permissão alguma.

Apagando um arquivo.

```
$ rm arquivo
```

Apagando um diretório.

```
$ rm -rf diretorio
```

**Note que:** O comando **rm -rf** deve ser usado com cuidado, pois após apagado um diretório e seus arquivos, não é possível recuperá-los.

- **cp** – O comando **cp** permite a cópia de arquivos e diretórios. Existem três formas básicas do comando: cópia de arquivos para arquivos, cópia de arquivos para diretórios e cópia de diretórios.

Copiando o arquivo /etc/passwd para o diretório corrente.

```
$ cp /etc/passwd
```

Copiando o diretório /etc para o diretório corrente com o nome de conf.

```
$ cp -R /etc ./conf
```

47

- **mv** – O comando **mv** permite a movimentação de um arquivo ou diretório de um local, no sistema, para outro. Apresenta três formas básicas: a primeira forma permite que se mude o nome do arquivo origem para o nome do arquivo destino (mesmo diretório); a segunda forma do comando faz a movimentação de todos os arquivos especificados para o diretório informado como destino e a terceira forma é similar à primeira, mas em vez de arquivo são utilizados diretórios.

Renomeando o arquivo passwd para senhas.txt.

```
$ mv passwd senhas.txt
```

Movendo o arquivo senhas.txt para o diretório conf.

```
$ mv senhas.txt conf
```

# Alterando Permissão dos Arquivos

O comando **chmod** permite alterar as permissões dos arquivos e diretórios. Somente o proprietário de um arquivo pode alterar suas permissões, mesmo que o grupo ou outros possuam direitos de escrita sobre o arquivo. O comando **chmod** tem a seguinte sintaxe:

## chmod [Opções] Permissões

A definição das permissões pode ser feita de forma simbólica ou octal (utiliza oito símbolos). A forma simbólica é a mais simples e, por isso, a mais usada por iniciantes. A forma octal é, no entanto, mais empregada, sobretudo em scripts antigos.

## Modo Simbólico

As permissões na forma simbólica têm a seguinte sintaxe:

[u g o a] [+ - =] [r w x u g o X]

As letras do primeiro grupo indicam de quem as permissões devem ser alteradas:

**u**: o usuário, proprietário do arquivo.

**g**: o grupo proprietário do arquivo.

**o**: outros (terceiros).

**a**: todos (*all*).

Os símbolos do segundo grupo indicam como os direitos devem ser alterados:

**+**: os direitos indicados devem ser adicionados.

**-**: os direitos indicados devem ser suprimidos.

**=**: os direitos devem ser ajustados ao valor indicado.

**r**: permissão de leitura.

**w**: permissão de escrita.

**x**: permissão de execução (ou acesso ao diretório).

**u**: usar as permissões atribuídas ao usuário proprietário.

**g**: usar as permissões atribuídas ao grupo proprietário.

**o**: usar as permissões atribuídas a outros.

Veja o exemplo a seguir:

Mostra os atributos iniciais.

```
$ ls -l
total 0
-rwxrw-rw- 1 marcos marcos 0 2009-12-10 16:49 diogo.c
-rwxrw-rw- 1 marcos marcos 0 2009-12-10 16:49 marcos.c
```

Retira de terceiros a permissão de escrita sobre todos os arquivos C no diretório corrente.

```
$ chmod o-w *.c  
$ ls -l  
total 0  
-rwxrw-r-- 1 marcos marcos 0 2009-12-10 16:49 diogo.c  
-rwxrw-r-- 1 marcos marcos 0 2009-12-10 16:49 marcos.c
```

Retira do grupo e de terceiros todas as permissões (leitura, escrita, execução) sobre todos os arquivos C no diretório corrente.

```
$ chmod go-rwx *.c  
$ ls -l  
total 0  
-rwx----- 1 marcos marcos 0 2009-12-10 16:49 diogo.c  
-rwx----- 1 marcos marcos 0 2009-12-10 16:49 marcos.c
```

Outro exemplo:

Mostra os atributos iniciais.

```
$ ls -l *.txt  
-rwxrwxrwx 1 marcos marcos 0 2009-12-10 17:01 instala.txt  
-r--rw-rw- 1 marcos marcos 0 2009-12-10 17:01 leiam.txt
```

Concede ao usuário permissão de escrita e ajusta ao grupo e outros somente permissão de leitura sobre os arquivos \*.txt do diretório corrente. Observe que as permissões podem ser agrupadas, usando vírgulas.

```
$ chmod u+w,go=r *.txt  
$ ls -l *.txt  
-rwxr--r-- 1 marcos marcos 0 2009-12-10 17:01 instala.txt  
-rw-r--r-- 1 marcos marcos 0 2009-12-10 17:01 leiam.txt
```

O comando **chmod** possui uma opção interessante **-R**, que permite atribuir permissões de maneira recursiva, ou seja, nos conteúdos dos subdiretórios. Assim, a melhor maneira de proteger seu diretório home dos olhares indiscretos de membros do seu grupo e de terceiros é executar o seguinte comando: **chmod -R go-rwx ~**

## Modo Octal

O uso do comando **chmod** em modo octal é similar ao modo simbólico, ou seja, são utilizados números ao invés de caracteres. As expressões de permissão são substituídas por valores octais representando as permissões desejadas. Assim, se desejarmos atribuir as permissões **rwxr-x---** a um arquivo **teste.c**, devemos converter o modo simbólico para o modo octal conforme segue:

r w x	r - x	- - -	expressão simbólica
1 1 1	1 0 1	0 0 0	em valores binários
7	5	0	em octal

Desta forma, deve ser executado o comando **chmod 750 teste.c** para obter as permissões desejadas.

A definição binária leva aos seguintes valores:

$$r = 4$$

$$w = 2$$

$$x = 1$$

67

Isto quer dizer que o valor 7 (rwx) é obtido pela soma de r + w + x ( $4 + 2 + 1$ ), o valor 5 (r - x) é obtido pela soma de r + x ( $4 + 1$ ) e finalmente o valor 0 (--) indica que não houve nenhuma soma.

A definição de permissões em modo octal é bem menos flexível que a notação simbólica, mas ainda muito usada, por ser aceita em todos os sistemas Linux. Além disso, sua compreensão é importante para o uso do comando **umask**.

 Cada número do conjunto octal indica que o perfil de um dos atributos (usuário, grupo ou terceiros) do arquivo está sendo modificado. O uso do formato octal é similar ao uso do formato simbólico ajustado.

Exemplo: **chmod 750 teste.c** é idêntico à

**chmod u=rwx,g=rx,o= teste.c.**

# Administração de Usuários

O Linux é um sistema operacional multiusuário, portanto é necessário que todos os usuários sejam cadastrados e tenham permissões de acesso diferenciadas. É possível, também, cadastrá-los em grupos, para facilitar o gerenciamento.

A criação e a administração de contas de usuários no sistema são exclusivas do superusuário (*root*). É uma tarefa que demanda responsabilidade e deve ser acompanhada com muita atenção.

81

## Verificando Informações do Usuário

Todo usuário possui um número chamado *user ID* com o qual o sistema Linux o identifica. Além do *user ID*, os usuários possuem o *group ID*. Toda vez que um processo for ativado será atribuído a este um *User ID* e um *Group ID*. Os ID's são chamados de identificação efetiva do processo.

Sintaxe: **id [ opções ][ nome ]**

Exemplo de uso:

Sem parâmetros, pegando as informações do usuário atual .

```
$ id  
uid=1000(marcos) gid=1000(marcos)  
grupos=4(adm),20(dialout),24(cdrom),46(plugdev),104(lpadmin),115()
```

Com parâmetros, pegando as informações do usuário bin.

```
$ id bin  
uid=2(bin) gid=2(bin) grupos=2(bin)
```

Pegando somente o user id do usuário atual.

```
$ id -u  
1000
```

## Tornando-se Outro Usuário – Comando su

Tal comando permite ao usuário mudar sua identidade, sem fazer o *logout*. É útil para executar um programa ou comando como superusuário sem ter que abandonar a seção atual.

Sintaxe: **su [-] [usuário]**

Em que:

- Quando informado, indica para iniciar as configurações do usuário que está sendo acessada.
- Usuário é o nome que a pessoa usa para acessar o sistema. Se não digitado, é assumido o usuário *root*.

Será pedida a senha do superusuário para autenticação. Digite **exit** quando desejar retornar à identificação do usuário anterior.

Exemplos de uso:

Acessando o usuário *root* sem carregar suas configurações.

Pegando as informações do usuário atual.

```
$ id  
uid=1000(marcos) gid=1000(marcos)  
grupos=4(adm),20(dialout),24(cdrom),46(plugdev),104(lpadmin),115()
```

Virando o usuário root.

```
$ su  
Senha:
```

Pegando as informações do usuário atual.

```
$ id  
uid=0(root) gid=0(root) grupos=0(root)
```

**Verificando que não houve mudança de usuário.**

```
$ pwd  
/home/marcos
```

**Saindo da conta root.**

```
$ exit  
exit
```

**Pegando as informações do usuário atual.**

```
$ id  
uid=1000(marcos) gid=1000(marcos)  
grupos=4(adm),20(dialout),24(cdrom),46(plugdev),104(lpadmin),115()
```

Acessando o usuário *root*, carregando suas configurações.

**Verificando o usuário atual.**

```
$ id  
uid=1000(marcos) gid=1000(marcos)  
grupos=4(adm),20(dialout),24(cdrom),46(plugdev),104(lpadmin),115()
```

**Tornando-se o usuário root e carregando suas configurações.**

```
$ su -  
Senha:
```

**Imprimindo o diretório atual e pegando as informações da nova conta.**

```
$ pwd  
/root  
$ id  
uid=0(root) gid=0(root) grupos=0(root)
```

**Finalizando a conta root.**

```
$ exit  
sair
```

**Imprimindo o diretório atual e pegando as informações conta.**

```
$ pwd  
/home/marcos  
$ id  
uid=1000(marcos) gid=1000(marcos)  
grupos=4(adm),20(dialout),24(cdrom),46(plugdev),104(lpadmin),115()
```

# Arquivo passwd e group

O arquivo **/etc/passwd** é o banco de dados dos usuários que podem logar no sistema. Tem formato de vários campos, os quais são separados pelo caractere : (dois pontos) e sempre na mesma ordem:

- Nome de *login* do usuário.
- Senha (criptografada).
- Id do usuário (identificação única, semelhante a um número de carteira de identidade).
- Grupo primário desse usuário (o usuário poderá participar de vários grupos).
- Nome completo (nome normal, sem ser o de *login*).
- Diretório home deste usuário.
- Shell inicial.

O exemplo a seguir mostra como são esses valores na prática:

```
marcos:x:1000:1000:Marcos Laureano,,,:/home/marcos:/bin/bash
```

84

Quando estamos utilizando **shadow password** (um pacote que evita o acesso de *hackers* ao conteúdo das senhas, mesmo criptografadas, dificultando, assim, a tentativa de quebra de senha), o segundo campo é substituído por um \* e a senha é armazenada em outro arquivo (**/etc/shadow**), normalmente inacessível.

O arquivo **/etc/group** define os grupos aos quais os usuários pertencem. Seu conteúdo são linhas da forma: **group\_name:passwd:GID:user\_list**, em que: **group\_name** – é o nome do grupo.

- **passwd** – um grupo pode opcionalmente ter uma senha.
- **GID (group id)** – é um código, como o *user id* (no arquivo **/etc/passwd**), mas relativo ao grupo.
- **user list** é a lista (separada por vírgulas) de todos os usuários que pertencem a este grupo.

Um usuário pode pertencer a qualquer número de grupos e herdará todas as permissões de acesso aos arquivos desses grupos.

# Adicionando Grupos – Comando groupadd

Para facilitar a administração do sistema, pode-se usar o conceito de grupos de usuários com perfis semelhantes. Por exemplo, definir grupos conforme os departamentos de uma empresa.

Sintaxe: **groupadd [ opções ] grupo**

Este comando irá alterar os arquivos:

- **/etc/group** – informações de grupos.
- **/etc/gshadow** – informações de grupos armazenadas de forma segura (senhas de grupo).

Criando o grupo de vendas e verificando o final do arquivo /etc/group.

```
$ groupadd vendas  
$ tail -3 /etc/group  
nogroup:x:65534:  
crontab:x:101:  
vendas:x:102:
```

Criando o grupo alunos com o Group ID = 2424.

```
$ groupadd -g 2424 alunos  
$ tail -3 /etc/group  
crontab:x:101:  
vendas:x:102:  
alunos:x:2424:
```

Criando o grupo teste sem informar o GID (observe que o sistema pega o último GID utilizado e soma 1).

```
$ groupadd teste  
$ tail -3 /etc/group  
vendas:x:102:  
alunos:x:2424:  
teste:x:2425:
```

# Eliminando Grupos – Comando groupdel

O comando **groupdel** permite que se eliminem grupos do sistema. Somente o superusuário poderá utilizá-lo.

Sintaxe: **groupdel grupo**

Exemplo de utilização:

Apagando os grupos criados anteriormente.

```
$ groupdel vendas  
$ groupdel alunos  
$ tail -3 /etc/group  
nogroup:x:65534:  
crontab:x:101:  
teste:x:2425:  
$ groupdel teste  
$ tail -3 /etc/group  
users:x:100:  
nogroup:x:65534:  
crontab:x:101:
```

86

# Adicionando Usuários – Comando useradd

O comando **useradd** permite que se criem usuários conforme especificado em opções. Somente o superusuário poderá utilizá-lo.

Sintaxe: **useradd [ opções ] usuário**

Este comando irá alterar os arquivos:

- **/etc/passwd** – informações de contas de usuários e senhas criptografadas.
- **/etc/shadow** – informações de contas de usuários e senhas criptografadas.
- **/etc/group** – informações de grupos.

Exemplos de utilização:

Criando o usuário Diogo, como não foi informado o grupo, o comando cria automaticamente um grupo com o mesmo nome do usuário.

```
$ useradd diogo  
$ id diogo  
uid=1000(diogo) gid=1000(diogo) groups=1000(diogo)  
$ tail -3 /etc/passwd  
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh  
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh  
diogo:x:1000:1000::/home/diogo:/bin/sh
```

Cria o usuário Rosa, especificando o seu grupo e o diretório home .

```
$ useradd rosa -d /home/rosa -g root  
$ id rosa  
uid=1001(rosa) gid=0(root) groups=0(root)  
$ tail -3 /etc/passwd  
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh  
diogo:x:1000:1000::/home/diogo:/bin/sh  
rosa:x:1001:0::/home/rosa:/bin/sh
```

87

## Alterando a Senha do Usuário – Comando passwd

O comando **passwd** permite que se troque a senha de determinado usuário. O superusuário pode trocar a senha de qualquer outro. O usuário comum, porém, pode trocar somente a sua senha. As senhas são armazenadas no arquivo **/etc/passwd** ou **/etc/shadow**. No arquivo **/etc/passwd** também são armazenadas as informações relativas aos usuários.

Após a criação do usuário será necessário criar uma senha para ele, caso contrário, não será permitido que o usuário faça *login* no sistema.

Sintaxe: **passwd [usuário]**

# Eliminando Usuários – Comando userdel

O comando **userdel** permite que se eliminem usuários do sistema. Somente o superusuário poderá utilizá-lo.

Sintaxe: **userdel [opções] usuário**

Exemplos de utilização:

Posiciona-se no diretório /home e lista os diretórios dos usuários.

```
$ cd /home  
$ ls -l  
total 8  
drwxr-xr-x 2 diogo diogo 4096 Dec 14 17:20 diogo  
drwxr-xr-x 2 rosa root 4096 Dec 14 17:20 rosa
```

Apaga o usuário Diogo, mas o seu diretório de usuário é mantido e fica órfão (sem dono).

```
$ userdel diogo  
$ ls -l  
total 8  
drwxr-xr-x 2 1000 1000 4096 Dec 14 17:20 diogo  
drwxr-xr-x 2 rosa root 4096 Dec 14 17:20 rosa
```

Apaga o usuário Rosa juntamente com o seu diretório de usuário.

```
$ userdel -r rosa  
$ ls -l  
total 4  
drwxr-xr-x 2 1000 1000 4096 Dec 14 17:20 diogo
```

**Atenção:** ao eliminar um usuário do sistema, verifique se em seu diretório home não há arquivos importantes. Na dúvida, elimine o usuário do sistema, mas mantenha o seu diretório home e peça para alguém analisar o conteúdo.

Assim que tiver eliminado um usuário, talvez seja interessante verificar se o UID antigo deste não tem propriedade sobre outros arquivos no sistema. Para localizar os caminhos de arquivos órfãos pode-se usar o comando **find** com o argumento **-nouser**.

## Você Sabia?

- A administração de usuários é muito importante em sistemas operacionais. Por questões de segurança a pessoa não pode utilizar senhas que possam ser facilmente descobertas e deve-se tomar cuidado para que um usuário não possa acessar os arquivos pessoais de outros usuários.
- Cada sistema operacional possui sua particularidade para o gerenciamento de usuário. Alguns sistemas operacionais possuem telas gráficas para facilitar o processo. No entanto, o princípio de gerência é o mesmo em todos os sistemas operacionais. São usuários ligados a grupos e permissões para usuários ou grupos de usuários (em segurança chamamos este tipo de controle de discricionário, pois ocorre a descrição das permissões). O sistema operacional Solaris (ex-Sun e atualmente pertencente a Oracle) implementa o conceito de administração de usuários baseado em papéis (*role based access control*) no qual, em vez de grupos se tem atribuições do que pode ser realizado no sistema operacional. Um usuário pode receber uma ou mais atribuições (papéis).



## Atividades

- 1) Crie os grupos aluno\_mesa01, aluno\_mesa02 e aluno\_mesa03.
- 2) Crie os usuários aluno\_01 e aluno\_02 com o grupo aluno\_mesa01.
- 3) Crie os usuários aluno\_03 e aluno\_04 com o grupo aluno\_mesa02.
- 4) Crie os usuários aluno\_05 e aluno\_06 com o grupo aluno\_mesa03.
- 5) Crie o grupo alunos\_geral.
- 6) Ligue todos os alunos\_0X ao grupo alunos\_geral.
- 7) Crie o usuário admina.
- 8) Ligue o usuário admina com os grupos aluno\_mesa01, aluno\_mesa02, aluno\_mesa03, root e alunos\_geral.
- 9) Mude o shell dos usuários ligados aos grupos aluno\_mesa01 e aluno\_mesa03 para /bin/sh.