Guia Foca GNU/Linux
Capítulo 13 - Permissões de acesso a arquivos e diretórios

A permissão de acesso protege o sistema de arquivos Linux do acesso indevido de pessoas ou programas não autorizados.

A permissão de acesso do `GNU/Linux` também impede que um programa mal intencionado, por exemplo, apague um arquivo que não deve, envie arquivos para outra pessoa ou forneça acesso da rede para que outros utilizadores invadam o sistema. O sistema `GNU/Linux` é muito seguro e como qualquer outro sistema seguro e confiável impede que utilizadores iniciantes (ou mal intencionados) instalem programas enviados por terceiros sem saber para que eles realmente servem e causem danos irreversíveis em seus arquivos, seu micro ou sua empresa.
Esta seção pode se tornar um pouco difícil de se entender, então recomendo ler e ao mesmo tempo prática-la para uma ótima compreensão. Não se preocupe, também coloquei exemplos para ajuda-lo a entender o sistema de permissões de acesso do ambiente `GNU/Linux`.

13.1 Donos, grupos e outros utilizadores

O princípio da segurança no sistema de arquivos `GNU/Linux` é definir o acesso aos arquivos por donos, grupos e outros utilizadores:

dono

É a pessoa que criou o arquivo ou o diretório. O nome do dono do arquivo/diretório é o mesmo do utilizador usado para entrar no sistema`GNU/Linux`. Somente o dono pode modificar as permissões de acesso do arquivo.

As permissões de acesso do dono de um arquivo somente se aplicam ao dono do arquivo/diretório. A identificação do dono também é chamada de user id (UID).

A identificação de utilizador e o nome do grupo que pertence são armazenadas respectivamente nos arquivos `/etc/passwd` e `/etc/group`. Estes são arquivos textos comuns e podem ser editados em qualquer editor de texto, mas tenha cuidado para não modificar o campo que contém a senha do utilizador encriptada (que pode estar armazenada neste arquivo caso não estiver usando senhas ocultas).

grupo

Para permitir que vários utilizadores diferentes tivessem acesso a um mesmo arquivo (já que somente o dono poderia ter acesso ao arquivo), este recurso foi criado. Cada utilizador pode fazer parte de um ou mais grupos e então acessar arquivos que pertençam ao mesmo grupo que o seu (mesmo que estes arquivos tenham outro *dono*).

Por padrão, quando um novo utilizador é criado, o grupo ele pertencerá será o mesmo de seu grupo primário (exceto pelas condições que explicarei adiante) (veja isto através do comando `id`, veja id, Seção 12.12). A identificação do grupo é chamada de `gid (group id)`.
Um utilizador pode pertencer a um ou mais grupos. Para detalhes de como incluir o utilizador em mais grupos veja Adicionando o utilizador a um grupo extra, Seção 12.10.

outros

É a categoria de utilizadores que não são donos ou não pertencem ao grupo do arquivo.

Cada um dos tipos acima possuem três tipos básicos de permissões de acesso que serão vistas na próxima seção.

13.2 Tipos de Permissões de acesso

Quanto aos tipos de permissões que se aplicam ao *dono*, *grupo* e *outros utilizadores*, temos 3 permissões básicas:

- `r` - Permissão de leitura para arquivos. Caso for um diretório, permite listar seu conteúdo (através do comando `ls`, por exemplo).
- `w` - Permissão de gravação para arquivos. Caso for um diretório, permite a gravação de arquivos ou outros diretórios dentro dele.

Para que um arquivo/diretório possa ser apagado, é necessário o acesso a gravação.

- `x` - Permite executar um arquivo (caso seja um programa executável). Caso seja um diretório, permite que seja acessado através do comando `cd` (veja cd, Seção 8.2 para detalhes).

As permissões de acesso a um arquivo/diretório podem ser visualizadas com o uso do comando `ls -la`. Para maiores detalhes veja ls, Seção 8.1. As 3 letras (rwx) são agrupadas da seguinte forma:

```
-rwxrwxrwx gleydson users teste
```

Virou uma bagunça não? Vou explicar cada parte para entender o que quer dizer as 10 letras acima (da esquerda para a direita):

- A primeira letra diz qual é o tipo do arquivo. Caso tiver um "d" é um diretório, um "l" um link a um arquivo no sistema (veja ln, Seção 10.4 para detalhes) , um "-" quer dizer que é um arquivo comum, etc.
- Da segunda a quarta letra (rwx) dizem qual é a permissão de acesso ao *dono* do arquivo. Neste caso *gleydson* ele tem a permissão de ler (r - read), gravar (w - write) e executar (x - execute) o arquivo `teste`.
- Da quinta a sétima letra (rwx) diz qual é a permissão de acesso ao *grupo* do arquivo. Neste caso todos os utilizadores que pertencem ao grupo *users* tem a permissão de ler (r), gravar (w), e também executar (x) o arquivo `teste`.
- Da oitava a décima letra (rwx) diz qual é a permissão de acesso para os *outros utilizadores*. Neste caso todos os utilizadores que não são donos do arquivo`teste` tem a permissão para ler, gravar e executar o programa.

Veja o comando chmod, Seção 13.7 para detalhes sobre a mudança das permissões de acesso de arquivos/diretórios.

13.3 Etapas para acesso a um arquivo/diretório

O acesso a um arquivo/diretório é feito verificando primeiro se o utilizador que acessará o arquivo é o seu *dono*, caso seja, as permissões de dono do arquivo são aplicadas. Caso não seja o *dono* do arquivo/diretório, é verificado se ele pertence ao grupo correspondente, caso pertença, as permissões do *grupo* são aplicadas. Caso não pertença ao *grupo*, são verificadas as permissões de acesso para os outros utilizadores que não são *donos* e não pertencem ao *grupo*correspondente ao arquivo/diretório.
Após verificar aonde o utilizador se encaixa nas permissões de acesso do arquivo (se ele é o *dono*, pertence ao *grupo*, ou *outros utilizadores*), é verificado se ele terá permissão acesso para o que deseja fazer (ler, gravar ou executar o arquivo), caso não tenha, o acesso é negado, mostrando uma mensagem do tipo: "Permission denied" (permissão negada).
O que isto que dizer é que mesmo que você seja o dono do arquivo e definir o acesso do *dono* (através do comando `chmod`) como somente leitura (r) mas o acesso dos *outros utilizadores* como leitura e gravação, você somente poderá ler este arquivo mas os outros utilizadores poderão ler/grava-lo.
As permissões de acesso (leitura, gravação, execução) para donos, grupos e outros utilizadores são independentes, permitindo assim um nível de acesso diferenciado. Para maiores detalhes veja Tipos de Permissões de acesso, Seção 13.2.

Lembre-se: Somente o dono pode modificar um arquivo/diretório!

Para mais detalhes veja os comandos chown, Seção 13.9 e chgrp, Seção 13.8.

13.4 Exemplos práticos de permissões de acesso

Abaixo dois exemplos práticos de permissão de acesso: Exemplo de acesso a um arquivo, Seção 13.4.1 e a Exemplo de acesso a um diretório, Seção 13.4.2. Os dois exemplos são explicados passo a passo para uma perfeita compreensão do assunto. Vamos a prática!

13.4.1 Exemplo de acesso a um arquivo

Abaixo um exemplo e explicação das permissões de acesso a um arquivo no `GNU/Linux` (obtido com o comando `ls -la`, explicarei passo a passo cada parte:

-rwxr-xr-- 1 gleydson user 8192 nov 4 16:00 teste

`-rwxr-xr--`

Estas são as permissões de acesso ao arquivo `teste`. Um conjunto de 10 letras que especificam o tipo do arquivo, permissão do dono do arquivo, grupo do arquivo e outros utilizadores. Veja a explicação detalhada sobre cada uma abaixo:

-rwxr-xr--

A primeira letra (do conjunto das 10 letras) determina o tipo do arquivos. Se a letra for um **d** é um diretório, e você poderá acessa-lo usando o comando `cd`. Caso for um **l** é um link simbólico para algum arquivo ou diretório no sistema (para detalhes veja o comando ln, Seção 10.4 . Um **-**significa que é um arquivo normal.

-**rwx**r-xr--

Estas 3 letras (da segunda a quarta do conjunto das 10 letras) são as permissões de acesso do *dono* do arquivo `teste`. O dono (neste caso*gleydson*) tem a permissão para ler (r), gravar (w) e executar (x) o arquivo `teste`.

-rwx**r-x**r--

Estas 3 letras (da quinta a sétima do conjunto das 10 letras) são as permissões de acesso dos utilizadores que pertencem ao *grupo user* do arquivo `teste`. Os utilizadores que pertencem ao grupo *user* tem a permissão somente para ler (r) e executar (x) o arquivo `teste` não podendo modifica-lo ou apaga-lo.

-rwxr-x**r--**

Estas 3 letras (da oitava a décima) são as permissões de acesso para utilizadores que **não** são *donos* do arquivo `teste` e que **não** pertencem ao grupo *user*. Neste caso, estas pessoas somente terão a permissão para ver o conteúdo do arquivo `teste`.

gleydson

Nome do dono do arquivo `teste`.

user

Nome do grupo que o arquivo `teste` pertence.

`teste`

Nome do arquivo.

13.4.2 Exemplo de acesso a um diretório

Abaixo um exemplo com explicações das permissões de acesso a um diretório no `GNU/Linux`:

drwxr-x--- 2 gleydson user 1024 nov 4 17:55 exemplo

`drwxr-x---`

Permissões de acesso ao diretório `exemplo`. É um conjunto de 10 letras que especificam o tipo de arquivo, permissão do dono do diretório, grupo que o diretório pertence e permissão de acesso a outros utilizadores. Veja as explicações abaixo:

drwxr-x---

A primeira letra (do conjunto das 10) determina o tipo do arquivo. Neste caso é um diretório porque tem a letra **d**.

d**rwx**r-x---

Estas 3 letras (da segunda a quarta) são as permissões de acesso do *dono* do diretório `exemplo`. O dono do diretório (neste caso *gleydson*) tem a permissão para listar arquivos do diretório (r), gravar arquivos no diretório (w) e entrar no diretório (x).

drwx**r-x**---

Estas 3 letras (da quinta a sétima) são as permissões de acesso dos utilizadores que pertencem ao *grupo user*. Os utilizadores que pertencem ao grupo *user* tem a permissão somente para listar arquivos do diretório (r) e entrar no diretório (x) `exemplo`.

drwxr-x**---**

Estas 3 letras (da oitava a décima) são as permissões de acesso para utilizadores que **não** são *donos* do diretório `exemplo` e que **não**pertencem ao grupo *user*. Com as permissões acima, nenhum utilizador que se encaixe nas condições de *dono* e *grupo* do diretório tem a permissão de acessa-lo.

gleydson

Nome do dono do diretório `exemplo`.

user

Nome do grupo que diretório `exemplo` pertence.

`exemplo`

Nome do diretório.

Para detalhes de como alterar o dono/grupo de um arquivo/diretório, veja os comandos chmod, Seção 13.7, chgrp, Seção 13.8 e chown, Seção 13.9.

OBSERVAÇÕES:

- O utilizador `root` não tem nenhuma restrição de acesso ao sistema.
- Se você tem permissões de gravação no diretório e tentar apagar um arquivo que você não tem permissão de gravação, o sistema perguntará se você confirma a exclusão do arquivo apesar do modo leitura. Caso você tenha permissões de gravação no arquivo, o arquivo será apagado por padrão sem mostrar nenhuma mensagem de erro (a não ser que seja especificada a opção -i com o comando `rm`).
- Por outro lado, mesmo que você tenha permissões de gravação em um arquivo mas não tenha permissões de gravação em um diretório, a exclusão do arquivo será negada.

Isto mostra que é levado mais em consideração a permissão de acesso do diretório do que as permissões dos arquivos e sub-diretórios que ele contém. Este ponto é muitas vezes ignorado por muitas pessoas e expõem seu sistema a riscos de segurança. Imagine o problema que algum utilizador que não tenha permissão de gravação em um arquivo mas que a tenha no diretório pode causar em um sistema mal administrado.

13.5 Permissões de Acesso Especiais

Em adição as três permissões básicas (rwx), existem permissões de acesso especiais (stX) que afetam arquivos executáveis e diretórios:

- `s` - Quando é usado na permissão de acesso do *Dono*, ajusta a identificação efetiva do utilizador do processo durante a execução de um programa, também chamado de *bit setuid*. Não tem efeito em diretórios.
  Quando `s` é usado na permissão de acesso do *Grupo*, ajusta a identificação efetiva do grupo do processo durante a execução de um programa, chamado de*bit setgid*. É identificado pela letra `s` no lugar da permissão de execução do grupo do arquivo/diretório. Em diretórios, força que os arquivos criados dentro dele pertençam ao mesmo grupo do diretório, ao invés do grupo primário que o utilizador pertence.
  Ambos *setgid* e *setuid* podem aparecer ao mesmo tempo no mesmo arquivo/diretório. A permissão de acesso especial `s` somente pode aparecer no campo*Dono* e *Grupo*.
- `S` - Idêntico a "s". Significa que não existe a permissão "x" (execução ou entrar no diretório) naquele lugar. Um exemplo é o chmod 2760 em um diretório.

- `t` - Salva a imagem do texto do programa no dispositivo swap, assim ele será carregado mais rapidamente quando executado, também chamado de *stick bit*.
  Em diretórios, impede que outros utilizadores removam arquivos dos quais não são donos. Isto é chamado de colocar o diretório em modo `append-only`. Um exemplo de diretório que se encaixa perfeitamente nesta condição é o `/tmp`, todos os utilizadores devem ter acesso para que seus programas possam criar os arquivos temporários lá, mas nenhum pode apagar arquivos dos outros. A permissão especial `t`, pode ser especificada somente no campo outros utilizadores das permissões de acesso.
- `T` - Idêntico a "t". Significa que não existe a permissão "x" naquela posição (por exemplo, em um chmod 1776 em um diretório).
- `X` - Se você usar `X` ao invés de `x`, a permissão de execução somente é afetada se o arquivo já tiver permissões de execução. Em diretórios ela tem o mesmo efeito que a permissão de execução `x`.
- Exemplo da permissão de acesso especial `X`:
- Crie um arquivo `teste` (digitando `touch teste`) e defina sua permissão para `rw-rw-r--` (`chmod ug=rw,o=r teste` ou `chmod 664 teste`).
- Agora use o comando `chmod a+X teste`
- digite `ls -l`

- Veja que as permissões do arquivo não foram afetadas.

- agora digite `chmod o+x teste`
- digite `ls -l`, você colocou a permissão de execução para os outros utilizadores.
- Agora use novamente o comando `chmod a+X teste`
- digite `ls -l`

- Veja que agora a permissão de execução foi concedida a todos os utilizadores, pois foi verificado que o arquivo era executável (tinha permissão de execução para outros utilizadores).

- Agora use o comando `chmod a-X teste`
- Ele também funcionará e removerá as permissões de execução de todos os utilizadores, porque o arquivo `teste` tem permissão de execução (confira digitando `ls -l`).
- Agora tente novamente o `chmod a+X teste`
- Você deve ter reparado que a permissão de acesso especial `X` é semelhante a `x`, mas somente faz efeito quanto o arquivo já tem permissão de execução para o dono, grupo ou outros utilizadores.
  Em diretórios, a permissão de acesso especial `X` funciona da mesma forma que `x`, até mesmo se o diretório não tiver nenhuma permissão de acesso (`x`).

13.6 A conta root

Esta seção foi retirada do Manual de Instalação da Debian.
A conta root é também chamada de *super utilizador*, este é um login que não possui restrições de segurança. A conta root somente deve ser usada para fazer a administração do sistema, e usada o menor tempo possível.

Qualquer senha que criar deverá conter de 6 a 8 caracteres (em sistemas usando crypto) ou até frases inteiras (caso esteja usando MD5, que garante maior segurança), e também poderá conter letras maiúsculas e

minúsculas, e também caracteres de pontuação. Tenha um cuidado especial quando escolher sua senha root, porque ela é a conta mais poderosa. Evite palavras de dicionário ou o uso de qualquer outros dados pessoais que podem ser adivinhados.

Se qualquer um lhe pedir senha root, seja extremamente cuidadoso. Você normalmente nunca deve distribuir sua conta root, a não ser que esteja administrando um computador com mais de um administrador do sistema.

Utilize uma conta de utilizador normal ao invés da conta root para operar seu sistema. Porque não usar a conta root? Bem, uma razão para evitar usar privilégios root é por causa da facilidade de se cometer danos irreparáveis como root. Outra razão é que você pode ser enganado e rodar um programa*Cavalo de Tróia* -- que é um programa que obtém poderes do *super utilizador* para comprometer a segurança do seu sistema sem que você saiba.

13.7 chmod

Muda a permissão de acesso a um arquivo ou diretório. Com este comando você pode escolher se utilizador ou grupo terá permissões para ler, gravar, executar um arquivo ou arquivos. Sempre que um arquivo é criado, seu dono é o utilizador que o criou e seu grupo é o grupo do utilizador (exceto para diretórios configurados com a permissão de grupo `"s"`, será visto adiante).
`chmod [`*`opções`*`] [`*`permissões`*`] [`*`diretório/arquivo`*`]`

Onde:

diretório/arquivo

Diretório ou arquivo que terá sua permissão mudada.

opções
-v, --verbose

Mostra todos os arquivos que estão sendo processados.

-f, --silent

Não mostra a maior parte das mensagens de erro.

-c, --change

Semelhante a opção -v, mas só mostra os arquivos que tiveram as permissões alteradas.

-R, --recursive
Muda permissões de acesso do *diretório/arquivo* no diretório atual e sub-diretórios.
ugoa+-=rwxXst

- *ugoa* - Controla que nível de acesso será mudado. Especificam, em ordem, utilizador (u), grupo (g), outros (o), todos (a).

- +-= - + coloca a permissão, - retira a permissão do arquivo e = define a permissão exatamente como especificado.
- rwx - *r* permissão de leitura do arquivo. *w* permissão de gravação. *x* permissão de execução (ou acesso a diretórios).

`chmod` não muda permissões de links simbólicos, as permissões devem ser mudadas no arquivo alvo do link. Também podem ser usados códigos numéricos octais para a mudança das permissões de acesso a arquivos/diretórios. Para detalhes veja Modo de permissão octal, Seção 13.10.
DICA: É possível copiar permissões de acesso do arquivo/diretório, por exemplo, se o arquivo `teste.txt` tiver a permissão de acesso `r-xr-----` e você digitar `chmod o=u`, as permissões de acesso dos outros utilizadores (o) serão idênticas ao do dono (u). Então a nova permissão de acesso do arquivo`teste.txt` será `r-xr--r-x`

Exemplos de permissões de acesso:

`chmod g+r *`

Permite que todos os utilizadores que pertençam ao grupo dos arquivos (g) tenham (+) permissões de leitura (r) em todos os arquivos do diretório atual.

`chmod o-r teste.txt`

Retira (-) a permissão de leitura (r) do arquivo `teste.txt` para os outros utilizadores (utilizadores que não são donos e não pertencem ao grupo do arquivo `teste.txt`).

`chmod uo+x teste.txt`

Inclui (+) a permissão de execução do arquivo `teste.txt` para o dono e outros utilizadores do arquivo.

`chmod a+x teste.txt`

Inclui (+) a permissão de execução do arquivo `teste.txt` para o dono, grupo e outros utilizadores.

`chmod a=rw teste.txt`

Define a permissão de todos os utilizadores exatamente (=) para leitura e gravação do arquivo `teste.txt`.

13.8 chgrp

Muda o grupo de um arquivo/diretório.

`chgrp [opções] [grupo] [arquivo/diretório]`

Onde:

grupo

Novo grupo do *arquivo/diretório*.

arquivo/diretório

Arquivo/diretório que terá o grupo alterado.

opções

-c, --changes

Somente mostra os arquivos/grupos que forem alterados.

-f, --silent

Não mostra mensagens de erro para arquivos/diretórios que não puderam ser alterados.

-v, --verbose

Mostra todas as mensagens e arquivos sendo modificados.

-R, --recursive

Altera os grupos de arquivos/sub-diretórios do diretório atual.

13.9 chown

Muda dono de um arquivo/diretório. Opcionalmente pode também ser usado para mudar o grupo.

`chown [opções] [dono.grupo] [diretório/arquivo]`

onde:

dono.grupo

Nome do *dono.grupo* que será atribuído ao *diretório/arquivo*. O grupo é opcional.

diretório/arquivo

Diretório/arquivo que o dono.grupo será modificado.

opções

-v, --verbose

Mostra os arquivos enquanto são alterados.

-f, --supress

Não mostra mensagens de erro durante a execução do programa.

-c, --changes

Mostra somente arquivos que forem alterados.

-R, --recursive

Altera dono e grupo de arquivos no diretório atual e sub-diretórios.

O *dono.grupo* pode ser especificado usando o nome de grupo ou o código numérico correspondente ao grupo (GID).

Você deve ter permissões de gravação no diretório/arquivo para alterar seu dono/grupo.

- `chown joao teste.txt` - Muda o dono do arquivo `teste.txt` para `joao`.
- `chown joao.users teste.txt` - Muda o dono do arquivo `teste.txt` para `joao` e seu grupo para `users`.
- `chown -R joao.users *` - Muda o dono/grupo dos arquivos do diretório atual e sub-diretórios para `joao/users` (desde que você tenha permissões de gravação no diretórios e sub-diretórios).

13.10 Modo de permissão octal

Ao invés de utilizar os modos de permissão `+r`, `-r`, etc, pode ser usado o modo octal para se alterar a permissão de acesso a um arquivo. O modo octal é um conjunto de oito números onde cada número define um tipo de acesso diferente.

É mais flexível gerenciar permissões de acesso usando o modo octal ao invés do comum, pois você especifica diretamente a permissão do dono, grupo, outros ao invés de gerenciar as permissões de cada um separadamente. Abaixo a lista de permissões de acesso octal:

- `0` - Nenhuma permissão de acesso. Equivalente a -rwx.
- `1` - Permissão de execução (x).
- `2` - Permissão de gravação (w).
- `3` - Permissão de gravação e execução (wx).
- `4` - Permissão de leitura (r).
- `5` - Permissão de leitura e execução (rx).
- `6` - Permissão de leitura e gravação (rw).
- `7` - Permissão de leitura, gravação e execução. Equivalente a +rwx.

O uso de um deste números define a permissão de acesso do *dono*, *grupo* ou *outros utilizadores*. Um modo fácil de entender como as permissões de acesso octais funcionam, é através da seguinte tabela:

```
1 = Executar

2 = Gravar

4 = Ler

* Para Dono e Grupo, multiplique as permissões acima por x100 e x10.
```

e para as permissões de acesso especiais:

```
1000 = Salva imagem do texto no dispositivo de troca

2000 = Ajusta o bit setgid na execução

4000 = Ajusta o bit setuid na execução
```

Basta agora fazer o seguinte:

- Somente permissão de execução, use 1.
- Somente a permissão de leitura, use 4.
- Somente permissão de gravação, use 2.
- Permissão de leitura/gravação, use 6 (equivale a 2+4 / Gravar+Ler).
- Permissão de leitura/execução, use 5 (equivale a 1+4 / Executar+Ler).
- Permissão de execução/gravação, use 3 (equivale a 1+2 / Executar+Gravar).
- Permissão de leitura/gravação/execução, use 7 (equivale a 1+2+4 / Executar+Gravar+Ler).
- Salvar texto no dispositivo de troca, use 1000.
- Ajustar bit setgid, use 2000.
- Ajustar bip setuid, use 4000.
- Salvar texto e ajustar bit setuid, use 5000 (equivale a 1000+4000 / Salvar texto + bit setuid).
- Ajustar bit setuid e setgid, use 6000 (equivale a 4000+2000 / setuid + setgid).

Vamos a prática com alguns exemplos:

```
"chmod 764 teste"
```

Os números são interpretados da **direita para a esquerda** como permissão de acesso aos *outros utilizadores* (4), *grupo* (6), e *dono* (7). O exemplo acima faz os *outros utilizadores* (4) terem acesso somente leitura (r) ao arquivo `teste`, o *grupo* (6) ter a permissão de leitura e gravação (w), e o *dono* (7) ter permissão de leitura, gravação e execução (rwx) ao arquivo `teste`.

Outro exemplo:

```
"chmod 40 teste"
```

O exemplo acima define a permissão de acesso dos *outros utilizadores* (0) como nenhuma, e define a permissão de acesso do *grupo* (4) como somente leitura (r). Note usei somente dois números e então a permissão de acesso do *dono* do arquivo não é modificada (leia as permissões de acesso da direita para a esquerda!). Para detalhes veja a lista de permissões de acesso em modo octal no inicio desta seção.

```
"chmod 751 teste"
```

O exemplo acima define a permissão de acesso dos *outros utilizadores* (1) para somente execução (x), o acesso do *grupo* (5) como leitura e execução (rx) e o acesso do *dono* (7) como leitura, gravação e execução (rwx).

"chmod 4751 teste"

O exemplo acima define a permissão de acesso dos *outros utilizadores* (1) para somente execução (x), acesso do *grupo* (5) como leitura e execução (rx), o acesso do *dono* (7) como leitura, gravação e execução (rwx) e ajusta o bit setgid (4) para o arquivo `teste`.

13.11 umask

A umask (*user mask*) são 3 números que definem as permissões iniciais do `dono`, `grupo` e `outros utilizadores` que o arquivo/diretório receberá quando for criado ou copiado. Digite `umask` sem parâmetros para retornar o valor de sua umask atual.
A umask tem efeitos diferentes caso o arquivo que estiver sendo criado for *binário* (um programa executável) ou *texto* (Arquivo texto e binário, Seção 2.2.3) . Veja a tabela a seguir para ver qual é a mais adequada a sua situação:

```
-------------------------------------------

| | ARQUIVO | DIRETÓRIO |

| UMASK |----------------------| |

| | Binário | Texto | |

|------------------------------|------------|

| 0 | r-x | rw- | rwx |

| 1 | r-- | rw- | rw- |

| 2 | r-x | r-- | r-x |

| 3 | r-- | r-- | r-- |

| 4 | --x | -w- | -wx |

| 5 | --- | -w- | -w- |

| 6 | --x | --- | --x |

| 7 | --- | --- | --- |

-------------------------------------------
```

Um *arquivo texto* criado com o comando `umask 012;touch texto.txt` receberá as permissões `-rw-rw-r--`, pois 0 (dono) terá permissões `rw-`, 1 (grupo), terá permissões `rw-` e 2 (outros utilizadores) terão permissões `r--`. Um *arquivo binário* copiado com o comando `umask 012;cp /bin/ls /tmp/ls` receberá as permissões `-r-xr--r-x` (confira com a tabela acima).

Por este motivo é preciso um pouco de atenção antes de escolher a umask, um valor mal escolhido poderia causar problemas de acesso a arquivos, diretórios ou programas não sendo executados. O valor padrão da umask na maioria das distribuições atuais é 022. A umask padrão no sistema Debian é a 022 .

A umask é de grande utilidade para programas que criam arquivos/diretórios temporários, desta forma pode-se bloquear o acesso de outros utilizadores desde a criação do arquivo, evitando recorrer ao `chmod`.