Permissões no GNU/Linux

Sistemas operacionais multi-usuário têm de zelar pela segurança e privacidade dos dados individuais de cada usuário, bem como prezar pela integridade do sistema.

Para isso existe as chamadas permissões de acesso, que atuam em dois aspectos fundamentais, o primeiro é a privacidade e o segundo, a segurança.

Quanto à privacidade, as permissões permitem que os usuários restrinjam o acesso a seus arquivos, caso esses possuam algum conteúdo confidencial ou algo parecido.

Você, como dono de um arquivo, pode especificar quem pode ler o seu arquivo, quem pode modificá-lo, apagá-lo ou executá-lo, tratando-se de um executável.

Quanto à segurança, um sistema nem sempre será utilizado somente por pessoas que possuem um bom conhecimento operacional.

Sendo assim, estarão sujeitas a cometer erros diversos, desde os mais banais até erros graves, como excluir arquivos essenciais para o funcionamento de alguns programas ou do sistema todo.

Assim, as permissões podem seguramente agir impedindo que determinados usuários possam excluir arquivos importantes ou mesmo executar programas que possam causar problemas ao sistema.

O *GNU/Linux* tem um método muito simples de lidar com permissões.
Inicialmente, elas são divididas em duas categorias: as permissões simples e as permissões especiais.

As permissões simples atuam liberando ou bloqueando o acesso à leitura, escrita e execução nos arquivos, existem diversas formas de se demonstrar as permissões de um arquivo.

Nesse instante mostraremos de forma simplificada, detalhes serão mostrados mais adiante:

Permissão	Literal	Octal
Leitura	r	4
Escrita	w	2
Execução	x	1

- Leitura (r): A permissão de leitura é a que vai dizer se o usuário tem ou não direito de ver o conteúdo do arquivo ou do diretório.

- Escrita (w): Essa permissão diz se o usuário terá ou não o direito de modificar o conteúdo do arquivo ou diretório.

- Execução (x): Por fim, a permissão de execução especifica se o usuário pode ou não executar o arquivo, caso ele se trate de um binário ou de um script. No caso de um diretório, especifica se o usuário poderá ou não acessá-lo.

Somente a existência dessas permissões não é suficiente.

Suponha que você crie um arquivo, e queira que o usuário joão tenha acesso a ele, mas nenhum outro usuário tenha.

Dessa forma, as permissões podem ser aplicadas ao:

- Dono: Chamamos de dono o usuário que criou o arquivo.
  O sistema de permissões no GNU/Linux permite que alteremos as permissões para nós próprios.
  Podemos, assim, evitar que, por exemplo, façamos alterações acidentais em arquivos importantes.

- Grupo: Como você sabe, todo usuário do sistema GNU/Linux pertence a pelo menos um grupo.
  Assim, você pode definir as permissões em nível de grupo, de forma que, se você liberar o acesso de leitura para o grupo professores, todos os usuários que fizerem parte desse grupo terão permissão de leitura no seu arquivo.

- Outros: Simplesmente, todos os usuários que não são você mesmo nem pertencem ao seu grupo primário.

USUÁRIO	PERMISSÕES		
	Leitura	Escrita	Execução
Dono	Sim/Não	Sim/Não	Sim/Não
Grupo	Sim/Não	Sim/Não	Sim/Não
Outros	Sim/Não	Sim/Não	Sim/Não

Obviamente, as permissões não são gravadas dessa forma complexa, mas de uma forma extremamente simples, que veremos mais adiante.

Podemos visualizar as permissões dos arquivos através do comando ls, utilizando o parâmetro -l, veja no exemplo abaixo a lista dos arquivos do diretório /home/davidson/:

```
$ ls -l /home/davidson
total 99373
-rwxr--r--  1 davidson davidson     6654 2005-03-07 05:27 apt-list
-rwxr--r--  1 davidson users    68591963 2005-02-15 05:52
backup.tar.gz
drwxr-xr-x  2 davidson davidson      264 2005-02-24 14:02 Desktop
drwxrwxrwx  5 davidson users         560 2005-03-01 14:31 doc
drwxrwxrwx  9 davidson users         432 2005-03-05 09:48 img
drwxr-xr-x  2 davidson davidson      136 2005-03-07 16:22 iso
-rw-r--r--  1 davidson davidson  2344858 2005-03-07 19:03 xine-
out.wav
```

As permissões, nesse caso, estão dispostas na primeira coluna.

Vamos analisar detalhadamente as permissões do arquivo backup.tar.gz:

```
$ ls -l backup.tar.gz
-rwxr--r--    1  davidson  users        68591963  2005-02-15  05:52
backup.tar.gz
```

Nessa sequência de caracteres (-rwxr--r--), o primeiro deles indica qual o tipo do arquivo.

Os caracteres exibidos podem ser:

- -: Arquivo regular;
- d: Diretório;
- b: Dispositivo de bloco;
- c: Dispositivo de caractere;
- p: FIFO;
- l: Link.

Os outros nove caracteres dividem-se em 3 grupos de 3 caracteres cada.

No primeiro deles, temos as permissões do dono do arquivo, no segundo, as permissões dos usuários que pertencem ao grupo primário do dono, e no terceiro, as permissões dos demais usuários.

Essas permissões estão dispostas da seguinte maneira:

- r / -: Permissão de leitura;
- w / -: Permissão de escrita;
- x / -: Permissão de execução.

Sendo assim, rwx significa que existe permissão de leitura (r), escrita (w) e execução (x).

Quando não há determinado tipo de permissão, o caracter correspondente é substituído por um – (hífen).

Dessa forma, r-x significa que existe permissão de leitura (r), não existe permissão de escrita (-) e há permissão de execução (x).

Veja abaixo a saída do comando ls -l, mostrando as permissões do arquivo backup.tar.gz:

$ ls -l backup.tar.gz

```
-rwxr--r--  1 davidson users    312 2005-03-29 15:27 backup.tar.gz
```

Através desse comando podemos ver as informações do arquivo tais como: permissões, dono, grupo, data de modificação e tamanho.

Permissão	Dono	Grupo	Arquivo
-rwxr--r--	davidson	users	backup

Na tabela acima, temos destacadas as principais informações.

As permissões estão dividas em quatro grupos de caracteres, explicados logo abaixo:

- -: O arquivo é um arquivo regular;
- rwx: O dono do arquivo tem permissão de leitura (r), escrita (w) e execução (x);
- r--: Os usuários que pertencem ao grupo do dono do arquivo têm permissão de leitura (r), não possuem permissões de escrita e execução (--);
- r--: Os demais usuários têm permissão de leitura (r), mas não possuem permissões de escrita e execução (--).

Dessa forma, qualquer usuário pode visualizar o conteúdo do arquivo backup.tar.gz, mas somente o seu dono poderá fazer alterações em seu conteúdo.

Definindo Permissões dos Arquivos

As permissões de um arquivo só podem ser alteradas por dois usuários: o administrador (root) e o dono do arquivo.

Para alterar as permissões, utilizaremos o comando chmod.

Sua sintaxe é a seguinte:

```
$ chmod [permissões] [arquivo]
```

Pode-se mudar as permissões de vários arquivos ao mesmo tempo, bastando listá-los separando-os por espaço.

As permissões podem ser definidas em dois formatos: literal e octal.

O Formato Literal

A sintaxe do formato literal é a seguinte:

```
[usuário][ação][permissão]
```

O usuário pode ser:

- u: O dono do arquivo (user);
- g: Os usuários que pertencem ao grupo do arquivo (group);
- o: Os outros usuários (others);
- a: Todos os usuários (all).

As ações podem ser:

- +: Adiciona a permissão;
- -: Remove a permissão;
- =: Iguala as permissões às especificadas.

E, por fim, as permissões:

- r: Permissão de leitura;
- w: Permissão de escrita;
- x: Permissão de execução.

Assim, consideremos novamente o arquivo backup.tar.gz:

```
$ ls -l backup.tar.gz
-rwxr--r--  1 davidson users 0 2005-03-09 10:58 backup.tar.gz
```

Podemos ver que os usuários que pertencem ao grupo users não possuem permissão de escrita, mas somente de leitura. Vamos, então, liberar a permissão de escrita para eles

```
$ chmod g+w backup.tar.gz
$ ls -l backup.tar.gz
-rwxrw-r--  1 davidson users 0 2005-03-09 10:58 backup.tar.gz
```

Agora as permissões de grupo são de leitura e escrita (rw-).

Suponhamos que este seja um arquivo confidencial, cujo conteúdo deve ficar restrito ao dono do arquivo e aos usuários que pertencem ao grupo users. Nesse caso, precisamos remover a permissão de leitura dos outros usuários.

Fazemos isso da seguinte forma:

```
$ chmod o-r backup.tar.gz
$ ls -l backup.tar.gz
-rwxrw----  1 davidson users 0 2005-03-09 10:58 backup.tar.gz
```

Agora, os outros usuários não possuem nenhuma permissão (---) sobre o arquivo backup.tar.gz.

Como você pode ver, o uso do chmod no formato literal é bem simples.

Mas há ainda dois pontos que podem ser destacados.

Primeiro, é que você definir mais de uma permissão por vez, colocando os caracteres r, w ou x em sequência.

Dessa forma, se você quisesse liberar para todos os usuários as permissões de leitura e execução no arquivo script.sh, o comando seria:

```
$ chmod a+rx script.sh
```

Outro ponto é que você pode aplicar diferentes permissões em uma única linha de comando, bastando separar os argumentos com vírgula.

Dessa forma, voltando ao exemplo anterior, para aplicar as permissões g+w e o-r em um único comando, bastaria fazer da seguinte forma:

```
$ chmod g+w,o-r backup.tar.gz
```

Por fim, além do + e do -, podemos utilizar o =. Quando usamos o =, precisamos informar todas as permissões.

Assim, para atribuir permissões de leitura e escrita para o grupo, utilizamos g=rw-, para atribuir permissão de leitura e execução para todos os usuários, usamos a=r-x, e assim por diante.

Formato Octal

As permissões no formato octal são escritas sob a forma de 3 números.

O primeiro número corresponde às permissões do dono do arquivo, o segundo número, às permissões do grupo, e o terceiro às permissões dos outros usuários.

Esses números, variam de 0 a 7, e são obtidos através da soma de outros números, cada um correspondendo a uma determinada permissão, a saber:

- 4: Permissão de leitura;
- 2: Permissão de escrita;
- 1: Permissão de execução.

Dessa forma, temos a seguinte lista de permissões no formato octal:

- 0: Nenhuma permissão;
- 1: Permissão de execução;
- 2: Permissão de escrita;
- 3: Permissão de execução e escrita;
- 4: Permissão de leitura;

- 5: Permissão de leitura e execução;
- 6: Permissão de leitura e escrita;
- 7: Permissão de leitura, escrita e execução.

Dessa forma, vamos considerar o arquivo script.sh.

Vamos definir suas permissões da seguinte forma:

- O dono do arquivo terá permissão total, ou seja, leitura, escrita e execução;
- O grupo do arquivo terá permissão de escrita e execução;
- Outros usuários terão permissão de leitura.

Essas permissões, no formato octal, são escritas como 754:

```
$ chmod 754 script.sh
$ ls -l script.sh
-rwxr-xr--  1 davidson users 0 2005-03-09 11:45 script.sh
```

A umask

Você deve ter percebido que todo novo arquivo recebe sempre as mesmas permissões, ainda que você nunca especifique-as no momento da criação e cópia dos arquivos, pois essas permissões são pré-definidas, e o programa que gerencia estas permissões é o umask.

Basicamente, a umask (user mask ou máscara de usuário) é constituído por 3 números que definem as permissões iniciais do dono, grupo e outros usuários para os arquivos que são criados ou copiados.

A umask tem efeitos diferentes para diretórios, arquivos binários e arquivos texto.

Para ver qual a umask definida no seu sistema, execute o comando umask:

```
$ umask
0022
```

O valor 0022 é o padrão, sendo o primeiro 0 apenas um indicador de que se trata de uma umask.

Os últimos 3 números correspondem às permissões umask para o dono, grupo e outros usuários.

	DIRETÓRIOS	ARQUIVOS	
		Texto	Binários
dono	rwx	rw-	r-x
grupo	r-x	r--	r-x
outros	r-x	r--	r-x

Vejamos um exemplo prático.

Suponha um diretório /tmp/teste/, contendo um diretório, um arquivo de texto e um arquivo binário, todos com permissão rwx para todos os usuários:

```
$ ls -l /tmp/teste
    -rwxrwxrwx  1 root root 10088 2005-03-09 17:37 arquivo_binário
    -rwxrwxrwx  1 root root  1506 2005-03-09 17:37 arquivo_texto
    drwxrwxrwx  2 root root    48 2005-03-09 17:37 dir
```

Agora, vamos copiar o diretório /tmp/teste/ para o nosso diretório pessoal, e ver o que acontece com as permissões dos arquivos.

```
$ cp -r /tmp/teste /home/davidson
$ ls -l /home/davidson/teste
    -rwxr-xr-x  1 davidson davidson 10088 2005-03-09 17:39 arquivo_binário
    -rwxr-xr-x  1 davidson davidson  1506 2005-03-09 17:39 arquivo_texto
    drwxr-xr-x  2 davidson davidson    48 2005-03-09 17:39 dir
```

Como podemos verificar, as permissões foram alteradas, e cada um dos arquivos recebeu permissões diferentes, de acordo com o tipo de cada um e obedecendo à umask 022.

Modificando a umask

Para modificar a umask do sistema, utilizamos o comando umask, segundo a sintaxe:

```
$ umask [modo]
```

Onde [modo] são os 3 números que definem as permissões de acordo com a tabela mostrada logo acima.

Como exemplo, vamos mudar o valor da umask para 247, depois recopiar o diretório /tmp/teste/ para /home/davidson/ e ver o que acontecem com as permissões:

```
$ umask 247
$ rm -rf /home/davidson/teste
$ cp -r /tmp/teste /home/davidson
$ ls -l /home/davidson/teste
    -r-x-wx---  1 davidson davidson 10088 2005-03-09 17:46 arquivo_binário
    -r-x-wx---  1 davidson davidson  1506 2005-03-09 17:46 arquivo_texto
    dr-x-wx---  2 davidson davidson    48 2005-03-09 17:46 dir
```

Como você pode ver, as permissões agora estão diferentes de antes.

Analise a tabela e procure entender porque as permissões ficam dessa forma quando se usa a umask 247.

Permissões especiais

Além das permissões simples de leitura, escrita e execução, existem outras três permissões, conhecidas como permissões especiais por só poderem ser aplicadas a tipos específicos de arquivos, que fornecem um alto nível de segurança ao sistema e privacidade aos usuários.

As permissões especiais são fornecidas pelos chamados bits, a saber: SUID, SGID e STICKY, dos quais trataremos a seguir.

SUID

O bit SUID (Set UID) pode ser aplicado somente para arquivos executáveis, para que eles rodem com as permissões do seu dono, e não com as permissões do usuário que o executou.

Dessa forma, suponha um executável que tenha sido criado pelo usuário root com o bit SUID ligado.

Então, se o usuário davidson executar esse arquivo, ele rodará como se fosse o root que o estivesse executando.

Isso pode ser útil para permitir que usuários comuns executem programas restritos ao administrador do sistema, como o shutdown, reboot, e outros.

Tomemos como exemplo o shutdown.

Ele está localizado no diretório /sbin/, e não pode ser executado por usuários comuns, retornando a seguinte mensagem de erro:

$ /sbin/shutdown

shutdown: you must be root to do that!

Ou seja, "shutdown: você precisa ser root para fazer isso!".

Vamos então ativar o buit SUID para permitir que o usuário davidson possa executar o shutdown:

 # chmod 4755 /sbin/shutdown

Ou

 # chmod u+s /sbin/shutdown

Observe que utilizamos 4 dígitos para o comando chmod no formato octal.

O primeiro dígito, que até então não tínhamos utilizado, é justamente aquele utilizado para as permissões especiais.

O número 4 corresponde ao bit SUID. Já no formato literal, utilizamos u+s para ativar (+) a permissão especial de execução (s) para os usuários (u).

Observe agora as novas permissões do /sbin/shutdown:

$ ls -l /sbin/shutdown

```
-rwsr-xr-x  1 root shutdown 18516 2005-01-04 19:43 /sbin/shutdown
```

Repare que agora existe um "s" no lugar do "x" nas permissões do dono.

Esse é o indicador de que o bit SUID está ligado.

Por fim, vamos criar um atalho para o shutdown no diretório /bin/, para que o usuário davidson possa executá-lo sem precisar digitar o caminho completo:

 # ln -s /sbin/shutdown /bin/shutdown

Agora, o usuário davidson e todos os demais usuários do sistema poderão executar normalmente o comando shutdown, como se fosse o root que o estivesse executando.

$ shutdown -h now

SGID

O bit SGID (Set GID), quando aplicado a arquivos executáveis, possui as mesmas funções do bit SUID, com a diferença de que as permissões são atribuídas ao grupo, e não ao usuário.

Ele deve ser usado para limitar o acesso a um executável do sistema a um determinado grupo.

Considerando o exemplo anterior, podemos criar um grupo chamado shutdown, e ativar o bit SGID de forma que somente os usuários que pertençam a esse grupo possam executar o utilitário shutdown:

```
# groupadd shutdown
# chown root:shutdown /sbin/shutdown
```

Os comandos acima criam o grupo shutdown e o definem como o grupo do utilitário shutdown.

Agora, para ativar o bit SGID, utilizamos o chmod na forma octal ou literal:

```
# chmod 2755 /sbin/shutdown
```

Ou

```
# chmod g+s /sbin/shutdown
```

No formato octal, veja que o número 2 corresponde ao bit SGID, enquanto os outros 3 números correspondem às permissões normais do arquivo.

Já no formato literal, utilizamos g+s para ativar (+) a permissão especial de execução (s) para o grupo (g).

As novas permissões do shutdown são agora:

```
# ls -l /sbin/shutdown
-rwxr-sr-x  1 root shutdown 18516 2005-01-04 19:43 /sbin/shutdown
```

Observe que o "x" das permissões do grupo foi substituída por um "s", indicando que o bit SGID está ligado.

Por fim, basta criar um link simbólico para o shutdown no diretório /bin/, para que os usuários possam executá-lo sem digitar o caminho completo.

```
# ln -s /sbin/shutdown /bin/shutdown
```

Agora, para permitir que algum usuário execute o shutdown, basta adicioná-lo ao grupo shutdown. Por exemplo, vamos adicionar o usuário davidson:

```
# addgroup shutdown davidson
```

Assim, o usuário davidson poderá executar o shutdown normalmente, mas os outros usuários não.

Sempre que se quiser que determinado usuário execute o shutdown, ele deverá ser adicionado ao grupo shutdown.

O bit SGID também pode ser aplicado a diretórios, e seu efeito neles é forçar os arquivos criados a pertencerem ao mesmo grupo do diretório pai, independente de quem seja o criador do arquivo.

Assim, suponha um diretório que pertença ao grupo vendas e possua o bit SGID ligado.

Então, todos os arquivos criados dentro desse diretório pertencerão ao grupo vendas, independente do grupo primário do dono do arquivo.

STICKY

O bit STICKY é utilizado em diretórios, para proteção de arquivos de uma forma menos radical, digamos assim.

Como você sabe, para evitar que outros usuários apaguem seus arquivos, basta colocá-los dentro de um diretório sem permissão de escrita para os outros usuários.

Entretanto, isso também faz com que os usuários não possam criar ou modificar nenhum arquivo dentro desse diretório, e é aí que o bit STICKY entra.

Suponha que um diretório chamado: documentos, possui o bit STICKY ligado, e com permissões de leitura e escrita para todos os usuários. Dessa forma, qualquer usuário pode criar arquivos dentro desse diretório.

Consideremos que os usuários davidson, fernando e frederico criaram os seguintes arquivos dentro desse diretório:

$ ls -l documentos

```
-rw-r--r--    1 davidson davidson 0 2005-03-11 10:43 debian-gnu-linux.sxw
-rw-r--r--  1 fernando fernando 0 2005-03-11 10:43 macro-linux.sxw
-rw-r--r--    1 frederico frederico 0 2005-03-11 10:43 software-livre.pdf
```

Os usuários podem então modificar todos esses arquivos.

Mas somente o usuário davidson pode excluir o arquivo debian-gnu-linux.sxw, somente o usuário fernando pode excluir o arquivo macro-linux.sxw e somente o usuário frederico pode excluir o arquivo software-livre.pdf.

Para ligar o bit STICKY num diretório, utilizamos o chmod no formato octal ou literal. Como exemplo, vamos ligar o bit STICKY no diretório documentos/.

$ chmod 1755 documentos

Ou

$ chmod o+t documentos

No formato octal, o número 1 corresponde ao bit STICKY, enquanto os outros 3 dígitos correspondem às permissões simples do diretório.

No formato literal, utilizamos o+t, para ativar (+) a restrição da exclusão de arquivos (t) para os outros usuários (o).

Observe agora as novas permissões do diretório documentos/:

$ ls -ld documentos

```
drwxr-xr-t  2 davidson davidson 160 2005-03-11 10:43 teste
```

Observe que o bit "x" das permissões dos outros usuários foi substituído por um "t", indicando que o bit STICKY está ligado.

Agora, basta dar permissão de escrita para os outros usuários, para que eles possam criar arquivos dentro desse diretório:

$ chmod o+w documentos

E pronto.

Qualquer usuário poderá criar arquivos dentro do diretório documentos/, mas os seus arquivos estarão a salvo, pois ninguém poderá exclui-los a não ser você próprio.

Fontes

http://www.guiafoca.org/cgs/guia/iniciante/ch-perm.html

http://www.vivaolinux.com.br/artigo/Tipos-de-permissoes-especiais-GNU-Linux

http://www.portugal-a-programar.pt/topic/3900-permissoes-em-gnulinux-linteam/

http://bespacoliberdade.wordpress.com/2011/10/18/entendendo-as-permissoes-de-arquivos-no-gnulinux/

http://wiki.nosdigitais.teia.org.br/Lista_de_Comandos_B%C3%A1sicos_GNU/Linux

http://jawakopi.wordpress.com/2010/03/12/gerenciamento-de-contas-e-permissoes-no-gnulinux/

http://www.slideshare.net/luiz_arthur/sistemas-operacionais-gnulinux-permisses-de-arquivos-diretrios

http://digitandocodigos.wordpress.com/2012/05/30/permissoes-de-arquivos-gnulinux/

São Paulo, SP, 29 de Martço de 2013

Mkmouse