Entendendo as Permissões de Arquivos no GNU/Linux

Mario Luiz Bernardinelli *(mariolb@gmail.com)

14 de Maio de 2009

Resumo

Cada sistema operacional possui características próprias e o entendimento de seu funcionamento permite que o usuário personalize o sistema de acordo com as suas necessidades. O grau de personalização depende é claro das características do sistema operacional utilizado.

Dentre as várias características existentes em cada sistema operacional, as permissões de acesso estão entre as mais importantes. Um simples atributo de arquivo pode significar proteger uma senha ou permitir que todos os usuários do sistema possam lê-la.

O Linux utiliza um esquema de permissões de acesso bastante simples, baseado basicamente em dono, grupo e outros, definindo assim que é o dono do arquivo (quem o criou), o grupo do arquivo e o que os demais usuários do sistema podem fazer com o arquivo. É sem dúvida um esquema bastante simples, herdado do sistema operacional Unix mas que, mesmo sendo simples, possui algumas características que podem confundir um pouco o usuário menos experiente, como é o caso dos bits SUID, SGID e Sticky. Entender a função e funcionamento destes atributos é essencial para uma boa administração do sistema, podendo inclusive ser utilizado para melhorar alguns aspectos de segurança do sistema operacional.

Palavras-chave: permissões de arquivos, linux.

1 Introdução

Todo arquivo digital possui duas partes: os dados e os metadados. Os dados correspondem ao conteúdo do arquivo propriamente dito e, normalmente é o que interessa ao usuário. Já os metadados correspondem aos atributos do arquivo no sistema de arquivos.

[1]Mario Luiz Bernardinelli é Tecnólogo em Processamento de Dados pela Faculdade de Tecnologia de Americana, possui os títulos de Especialista em Engenharia de Sotware e Especialista em Redes de Computadores, ambos pela Unicamp, e as certificações Linux LPI-C1 e LPI-C2. Atualmente, trabalha como desenvolvedor de software em ambiente Linux e administrador de redes para uma empresa do ramo de sistemas de automação.

Dentre estes atributos os de conhecimento mais comum são o nome do arquivo e a data do último acesso. Porém, cada sistema de arquivos pode associar diferentes conteúdos para os metadados de um arquivo.

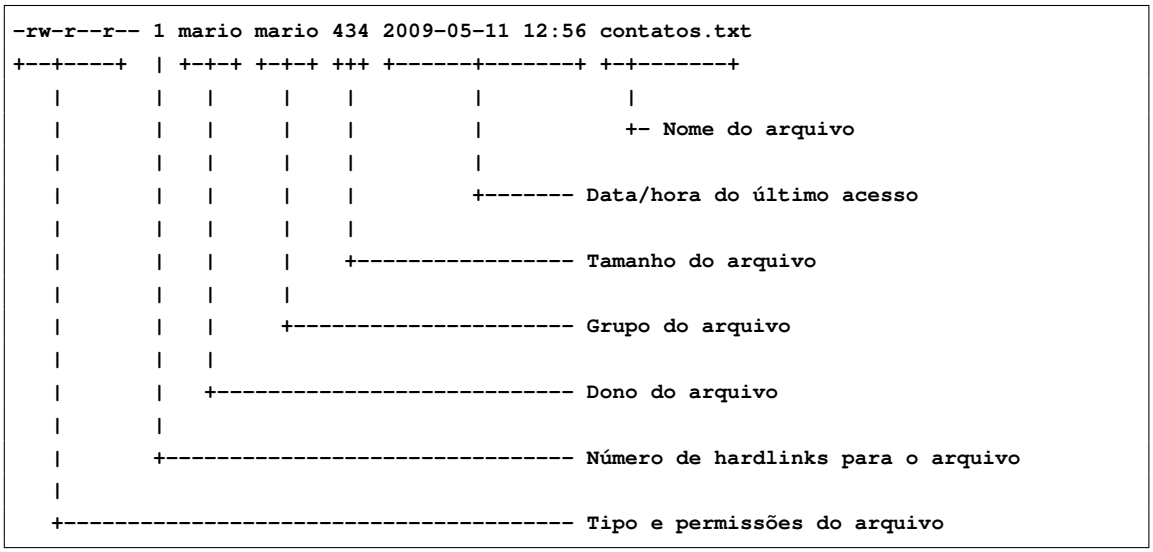
Neste artigo serão apresentadas em linhas gerais as permissões dos arquivos em sistemas operacionais Linux usando sistemas de arquivos padrões do Linux, como ext3, ext4, reiserfs etc.

2 Entendendo as permissões de arquivos

No Linux, quando listamos os arquivos de um diretório, podem ser mostrados vários atributos dos arquivos, além dos nomes. Por exemplo, o comando ls -l lista as permissões de acesso, o número de hardlinks, o nome do dono e do grupo do arquivo, o tamanho, a data do último acesso e o nome do arquivo. Veja o exemplo:

```
mario@quark ~$ ls -l
total 0
-rw-r--r-- 1 mario mario 434 2009-05-11 12:56 contatos.txt
```

Pode não parecer, mas dá para saber muita coisa do arquivo somente olhando para esta linha. Vejamos:



Para o objetivo deste artigo apenas alguns campos são interessantes, mas vamos descrever brevemente o significado de todos eles.

Os três últimos (nome, data/hora e tamanho) são auto-explicativos.

O segundo campo (número de hardlinks) indica, basicamente, o número de referências existentes para o arquivo.

O terceiro e quarto campos são importantes porque indicam o nome do usuário e o grupo associados ao arquivo, enquanto o primeiro campo define o tipo do arquivo e as permissões de acesso do dono do arquivo, do grupo do arquivo e dos demais usuários do sistema.

O primeiro campo exige algumas considerações: o primeiro caractere identifica o tipo do arquivo, a saber:

- Arquivo regular

d Diertório

l Link simbólico

b Dispositivo de bloco (um disco rígido, por exemplo)

c Dispositivo de caractere (por exemplo, um terminal ou a porta serial do sistema)

p Pipe (FIFO)

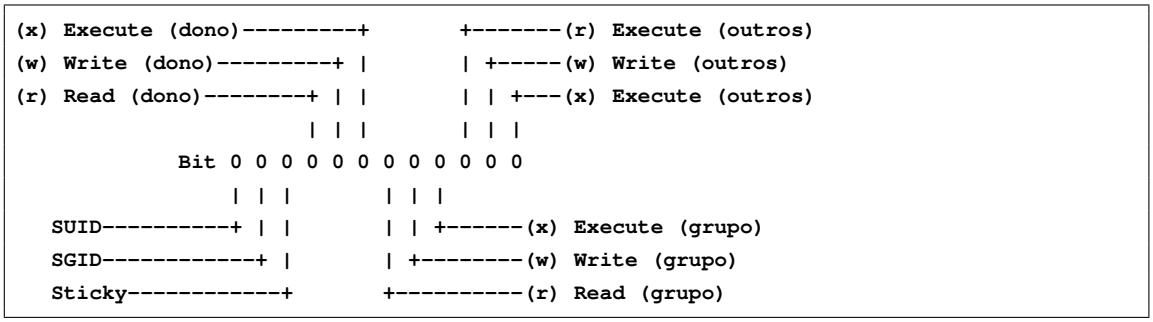
s Socket (Unix socket)

Ainda no primeiro campo, os demais caracteres representam as permissões de acesso ao arquivo. São nove caracteres que devem ser agrupados em grupos de três, conforme exemplo a seguir:

```
-rwxrwxrwx
 | || || |
 | || |+-+---- Permissões do todos os usuários
 | || |
 | |+-+------- Permissões do grupo, conforme especificado no
 | |
 +-+---------- Permissões do dono do arquivo
```

Observe que este grupo de caracteres especifica apenas as permissões, mas não identifica o dono ou o grupo. A identificação do dono e do grupo aparecem no terceiro e quarto campos, respectivamente.

As permissões de arquivos são definidas sob a forma de 12 bits, sendo que cada bit identifica uma única permissão. O comando 'ls -l' mostra apenas 9 caracteres (inves de 12 como poderia ser esperado). Por enquanto, vamos considerar os 12 bits para entendermos o que cada um significa:



Inicialmente, vamos pensar apenas no dono, grupo e outros, que correspondem aos últimos bits da máscara de permissões. Observe que as permissões para estes três objetos se repetem, isto é, podem ser leitura (r), escrita (w) e execução (x). Qualquer combinação destas permissões é aceita, o que não significa que farão algum sentido...

As permissões de leitura (read) e gravação (write) dispensam comentários. A permissão de execução (execute) merece uma pequena atenção: se o objeto for um arquivo, então se ele for acionado (com um duplo clique com o mouse numa interface gráfica ou através de uma chamada em linha de comando, por exemplo), o sistema operacional irá tentar executá-lo. Por outro lado, se o objeto for um diretório, a permissão de execução (execute) define se o diretório pode ou não ser acessado.

Os bits SUID, SGID e Sticky são permissões especiais. Quando um programa é executado, ele tem as mesmas permissões do usuário que o executou. Assim, por exemplo, se o usuário marcos executar o comando (programa) cat, este comando será executado com os mesmos privilégios do usuário marcos. Esta característica é desejável para a maioria das aplicações. No entanto, há casos em que é necessário que os programas sejam executados com privilégios especiais. É o caso, por exemplo, do comando passwd. Este comando permite que o usuário troque a sua senha. No Linux, via de regra, os hashes das senhas dos usuários são armazenadas no arquivo /etc/shadow. Este arquivo, por questões de segurança, só pode ser acessado pelo administrador do sistema (root), evitando que os usuários consigam obter os hashes das senhas de outros usuário (inclusive do root) e executar ataques de força bruta para obter a senha dos mesmos.

Vejamos as permissões do arquivo /etc/shadow:

```
mario@quark ~$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1388 2009-05-12 12:43 /etc/shadow
```

Observe as permissões (lembre-se que o primeiro caractere indica o tipo do arquivo e os outros nove carateres devem se agrupados em grupos de três caracteres):

- O dono do arquivo (root) pode ler e escrever no arquivo (rw-)

- O grupo (shadow) pode apenas ler o arquivo (r- -)
- Os demais usuários (outros) não podem acessar o arquivo (- - -)

Neste cenário, temos um problema: como o usuário poderia trocar a sua senha se ele não pode nem ler o arquivo /etc/shadow? Para resolver este problema foi criado o bit SUID: se habilitado, o SUID bit informa ao sistema operacional que o programa deve ser executado com os mesmos privilégios do dono do arquivo, não importando que usuário executou o programa. O programa que permite a alteração da senha é o passwd. Vejamos os privilégios do mesmo:

```
mario@quark ~$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 28480 2009-01-14 11:36 /usr/bin/passwd
```

Observe que o dono do arquivo é o root e que as suas permissões são as seguintes: rws. Lembra-se quando eu disse que as permissões dos arquivos correspondem a doze bits que ligam ou desligam as flags de permissões? Pois é, o comando ls mostra os três bits especiais (SUID, SGID e Sticky) junto com as permissões do dono, grupo e outros. Observe o caractere 's' nas permissões do arquivo passwd: ele indica que o bit SUID está ligado. Em outras palavras, quando um usário executar o programa passwd, ele será executado como se fosse o usuário root (já que as permissões 'rws' são do dono do arquivo que, no caso do arquivo passwd é o root).

Com isso, o problema de troca de senhas é resolvido: qualquer usuário pode trocar a sua senha e como um programa passwd legítimo só permite a troca de senhas e nenhuma informação de outro usuário é utilizada ou mostrada, o processo é seguro (desde que que o programa passwd seja legítimo).

Há outros casos de uso do SUID bit, como o comando ping que precisa criar pacotes IP especiais (chamados raw) e também alguns programas do ambiente gráfico que precisam acesso privilegiado a alguns dispositivos.

Porém, apesar de ter sua função no sistema, o uso do SUID bit deve realizado com critério, pois o seu uso indiscriminado pode afetar seriamente a segurança do sistema. Na minha opinião, com os avanços do Linux no terreno dos desktops, algumas distribuições, senão todas, têm habilitado por padrão o SUID bit de forma crescente em muitos programas com a finalidade de facilitar a vida e melhorar a usabilidade do sistemas para os usuários sem grandes conhecimentos técnicos. Acho que esta prática pode prejudicar a segurança do sistema operacional se for conduzida de forma indiscriminada.

O bit SGID tem a mesma função que o SUID bit, exceto que o arquivo é executado com os mesmos privilégios do grupo do arquivo. Além disso, o SGID tem um significado especial quando é usado em diretórios: uma vez habilitado o SGID num diretório,

todos os novos arquivos criados neste diretório terão como grupo o grupo atribuído ao diretório, independente do grupo do usuário que criou o arquivo. Por exemplo, se o diretório /home/desenv for do grupo desenvolvimento e o bit SGID estiver habilitado, todos os arquivos criados dentro deste diretório serão criados com o grupo desenvolvimento, independente do grupo do usuário que criou o arquivo.

O último bit do grupo de permissões especiais é o Sticky. Antigamente este bit era utilizado em determinados programas executáveis para informar ao sistema operacional para que mantivesse uma imagem do programa em memória após o término do mesmo. Esta capacidade melhorava a resposta do sistema em chamadas subsequentes do programa pois eliminava a fase de carregamento do programa do disco. Esta característica era utilizada em programas grandes que demoravam muito para carregar ou em programas executados com muita frequência. O uso de modernas técnicas de memória virtual tornaram desnecessário o uso desta característica.

Quando aplicado em diretórios, o bit Sticky oferece um recurso especial para os arquivos criados dentro dos diretórios: independente das permissões do arquivo, o únicos usuários que podem apagar ou renomear o arquivo é o próprio dono do arquivo, o dono do diretório e o root. Um exemplo prático do uso do bit Sticky é o diretório /tmp: neste diretório, todos os arquivos são acessíveis por todos os usuários, porém, somente o dono do arquivo e o root podem removê-los ou renomeá-los.

```
mario@quark ~$ ls -ld /tmp
drwxrwxrwt 15 root root 592 2009-05-14 08:13 /tmp
```

Observe o caractere 't' nas permissões globais: ele indica que o sticky bit está ativado.

3 Conclusão

O entendimento do funcionamento das permissões de arquivos em qualquer sistema operacional é muito importante. A partir deste entendimento é que boa parte do sistema pode ser adequada ao uso desejado. Até mesmo a segurança do sietema pode ser melhorada se o funcionamento das permissões de arquivos for bem compreendida e aplicada.

Com a popularização dos computadores, cada vez mais os sistemas operacionais são projetados tendo em vista que o usuário final pode não ter os conhecimentos necessários para adequar o sistema às suas necessidades. O ponto forte desta visão é que os sistemas estão ficando cada vez mais fáceis de usar. Em contrapartida, tornar o sistema fácil de usar implica normalmente em estender os privilégios dos usuários ou dos programas por eles utilizados, muitas vezes euiparando-os aos privilégios do administrador

do sistema. Como um grande poder implica numa grande responsabilidade, atribuir ao usuário final, que pode ser leigo no assunto, um grande poder, trás sérias implicações para a segurança do sistema.