Administração de usuários
Permissões

Administração de usuários - Permissões

Introdução

As **permissões** do GNU/Linux são mecanismos que permitem que um usuário restrinja o acesso a um arquivo ou diretório no sistema de arquivos. Para um **arquivo**, um usuário pode especificar quem pode **ler, escrever** e **executar** (no caso de programas executáveis de Shell Script). Para os **diretórios**, um usuário pode especificar quem pode **ler o diretório** (listar seu conteúdo), **escrever nele** (adicionar ou remover arquivos ou pastas) e **entrar no diretório**.

Tipos de permissões

O status de permissão de cada arquivo é expresso em **tokens**. Os tokens de permissão são:

r – Acesso de Leitura

w – Acesso de Gravação

x – Acesso de Execução

Administração de usuários - Permissões

Visualizando as permissões

Para determinar permissões em um arquivo ou diretório, liste-os em formato longo utilizando o comando **ls -l**:

 # ls -l /home/leo

-rw-r--r-- 1 leo leo 7502 Mar 15 12:00 carta.txt

1	2 3 4	5 6 7	8 9 10
-	rw-	r--	r--
Diretório (**d**) ou Arquivo (**-**)	Permissões do proprietário	Permissões do grupo	Permissões outros

Administração de usuários - Permissões

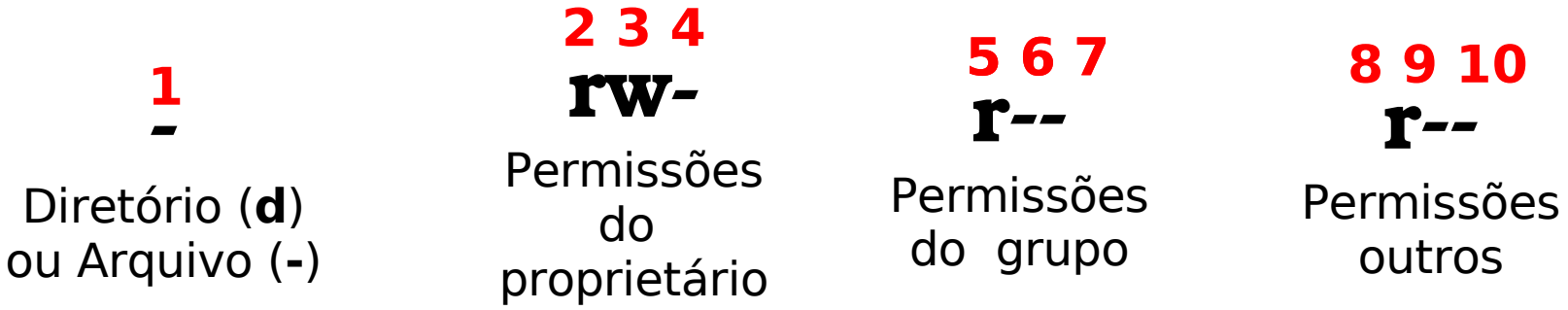
Visualizando as permissões

Aqui trazemos o significado de cada um dos campos das permissões vistas com o comando **ls -l**, se forem aplicadas em um arquivo ou diretório:

Campo 1 - especifica o tipo de recurso:

- => representa um arquivo
d => representa um diretório
l => representa um link simbólico
b => dispositivo de bloco
c => dispositivo de caracteres

Administração de usuários - Permissões



Campos 2, 3 e 4 - Permissões do **Proprietário**: essas permissões mostram os direitos que o proprietário tem sobre o arquivo ou diretório.

Campos 5, 6 e 7 - Permissões do **Grupo**: essas permissões mostram o direito de acesso de todos os usuários pertencentes ao grupo.

Campos 8, 9 e 10 - Permissões **Outros**: essas permissões mostram quais direitos, se houver algum, todos os demais usuários têm para acessar esse arquivo/diretório.

Administração de usuários - Permissões

chmod

Modifica as permissões de um arquivo ou diretório. Você precisa ser **proprietário** do arquivo/diretório, ou ter acesso de **root**, para modificar permissões.

O comando **chmod** pode ser usado para mudar os tokens "**rwx**" dos arquivos e/ou diretórios, a sintaxe básica do comando é:

 # chmod <usuário**>+<**tokens**> <**arquivo/diretório**> (adiciona permissão)**

 # chmod <usuário**>-<**tokens**> <**arquivo/diretório**> (remove permissão)**

 # chmod <usuário**>=<**tokens**> <**arquivo/diretório**> (atribui permissão)**

Exemplo:

Se eu quero mudar a permissão para o dono do arquivo (**u=user**) poder ler e gravar (**rw**) no **arquivo.txt**, faço o seguinte:

$ chmod u=rw arquivo.txt

Administração de usuários - Permissões

chmod

Caso você queira tirar a permissão de gravação e leitura do dono do arquivo:

$ chmod u-rw arquivo.txt

Assim é dada permissão de gravação para todos usuários do sistema:

$ chmod o+w arquivo.txt

Administração de usuários - Permissões

Usando chmod com o método octal

Usar o **chmod** com o **método octal** é bastante prático. Em vez de usar letras como símbolos para cada permissão, usa-se números.

OCTAL	PERMISSÃO	CÁLCULO
0	---	0
1	--x	1
2	-w-	2
3	-wx	2+1=3
4	r--	4
5	r–x	4+1=5
6	rw-	4+2=6
7	rwx	4+2+1=7

Administração de usuários - Permissões

chmod

Você precisa ser proprietário do arquivo/diretório, ou ter acesso de root, para modificar as suas permissões.

Exemplo:

 # chmod 777 4linux.txt

Outra maneira de dar a permissão acima:

 # chmod u=rwx,g=rwx,o=rwx 4linux.txt (forma literal)

Neste exemplo eu estou atribuindo a permissão 777 para o arquivo 4linux.txt, ou seja, como na tabela anterior estou atribuindo rwx ao dono, ao grupo e aos outros.

Com o comando **ls -l** podemos ver as permissões atribuídas:

-rwxrwxrwx 3 4linux 4linux 4096 May 6 17:14 01 4linux.txt

Administração de usuários – Permissões especiais

chmod

Existem casos especiais que veremos em chat, que precisamos usar as permissões especiais. Isso vai ficar totalmente claro em chat! **:-)**

Tokens especiais:

s => se este token estiver ligado nas **permissões do dono (u)** , o arquivo é executado como se fosse pelo dono, não faz sentido em diretórios (podemos dar poder a um usuário comum para executar o comando **shutdown** por exemplo);

s => se este token estiver ligado nas **permissões do grupo (g)**, o arquivo poderá ser executado como se fosse invocado por um membro do grupo proprietário; todo arquivo criado em um diretório com esse token ligado é criado com o mesmo grupo do diretório **(veremos a importância disso no chat)**.

Administração de usuários – Permissões especiais

chmod

t => um arquivo criado com esse token ligado só pode ser apagado pelo seu proprietário, mesmo se esse arquivo estiver num diretório de um grupo que compartilha desse mesmo arquivo (essa permissão é utilizada no diretório **/tmp**).

Esses **tokens** têm outros nomes e podem ser representados na forma octal também.

s para usuário é chamado de **setuid**;

s para grupo é chamado de **setgid;**

t é chamado de **bit stick**;

Os números também possuem nomes:

suid-bit = 4

sgid-bit = 2

stick-bit = 1

Administração de usuários – Permissões especiais

chmod

E para dar ou tirarmos essas permissões também usamos o comando **chmod**.

Para isto, basta "acrescentar" o número correspondente a permissão especial dada.

Exemplos:

 # chmod 4700 /sbin/shutdown => para dar permissão **suid-bit** ao arquivo **/sbin/shutdown**; (falta detalhes que mostrarei em chat para funcionar para usuário normal);

 # chmod 2770 arquivos => para dar permissão de **sgid-bit** ao diretório arquivos (assim os arquivos criados dentro desse diretório recebe o grupo do mesmo, usado em diretórios públicos);

 # chmod 1770 arquivos => para dar permissão de **stick-bit** ao diretório arquivos (assim só o dono do arquivo remove o mesmo).

Administração de usuários – Permissões especiais

SUID-BIT

Essa permissão é aplicada apenas em arquivos executáveis e não é muito recomendada pois pode dar poder a quem não deve! Ela permite que qualquer usuário execute o arquivo como se fosse dono deste arquivo.

Exemplo:

 # chmod 4755 script.sh

$ ls -l /home/ana

-rwsr-xr-x 1 ana ana 657 Dec 14 15:04 script.sh

$ whoami

gaby

$ ./script.sh

Administração de usuários – Permissões especiais

SGID-BIT

Permissão dada apenas em diretórios. Quando você grava um arquivo, o grupo dono desse arquivo (independente de onde você criou) é o seu grupo primário. Porém, **se no diretório,** que você criou esse arquivo, tiver a permissão **sgid-bit**, o grupo dono do arquivo será o grupo dono do diretório corrente, e não o seu grupo pessoal.

Essa permissão é muito usada para compartilhar arquivos de uma determinada pasta entre usuários que pertençam ao mesmo grupo.

 # chmod 2775 hackerteen

 # ls -l /mnt

drwxrw**s**r-x 1 root **ht** 657 Dec 14 15:04 **hackerteen**

 # ls -l /mnt/hackerteen

-rwxrwxr-x 1 ana **ht** 657 Dec 14 15:04 **script.sh**

Administração de usuários – Permissões especiais

STICK-BIT

Essa permissão trabalha junto com o **sgid-bit**, garantindo segurança para ela. Também é dada apenas em diretórios, um exemplo é o **/tmp** que é público. Quando você dá a permissão sgid-bit esses usuários têm a mesma permissão dada ao grupo. Se a permissão para o grupo é de leitura, gravação e execução, todos os usuários do grupo podem mover, renomear e até mesmo remover arquivos de outro usuário que pertença a esse grupo. Daí a permissão stick-bit garante que os arquivos do diretório com permissão sgid-bit fiquem seguros, isto é, nenhum usuário que não seja o dono do arquivo possa mover/renomear ou remover os arquivos. **Exemplo:**

 # chmod 3775 hackerteen => aqui eu defino permissão sgid-bit e também stick-bit para o diretório hackerteen (**2+1=3**)

 # ls -l /mnt

drwxrws--t 1 root ht 657 Dec 14 15:04 hackerteen

Administração de usuários

chown

Muda o dono e pode mudar grupo de um arquivo.

Exemplos:

 # chown leonardo.instrutores apostila.doc

O dono do arquivo **apostila.doc** passa ter como dono o usuário **leonardo** e o grupo passa a ser **instrutores**.

Se eu quiser mudar apenas o dono:

 # chown leonardo apostila.doc

Se formos mudar o dono de um diretório e quisermos que essa alteração afete também todos os subdiretórios e arquivos dele, usamos uma opção(**-R** de recursivo) a mais:

 # chown -R leonardo.instrutores apostila.doc

Administração de usuários

chgrp

Modifica o identificador de grupo ("**group ID**", **GID**) dos arquivos passados como argumentos. O GID pode ser um número decimal especificando o group id, ou o nome do grupo encontrado no arquivo **/etc/group**.

Exemplo:

 # chgrp 1000 nome_arquivo

ou

 # chgrp diretoria nome_arquivo

ou

 # chgrp -R diretoria nome_arquivo (recursivamente)

Nota: Você deve ser o proprietário do arquivo, ou o superusuário, para que possa utilizar este comando.

Bibliografia

Linux – Guia do Administrador do Sistema
Autor: Rubem E. Pereira
Editora: Novatec

Manual Completo do Linux (Guia do Administrador)
Autor: Evi Nemeth, Garth Snyder, Trent R. Hein
Editora: Pearson Books

Guia Foca GNU/Linux
http://focalinux.cipsga.org.br/