UNIX: Permissões de acesso em arquivos

Definições básicas

O UNIX possui um sistema de controle de acesso ao sistema de arquivos seguindo o paradigma de Listas de Controle de Acesso (ACL - *Access Control Lists*). A cada arquivo ou diretório são associados:

- Um **usuário** proprietário (*owner*). Normalmente é quem criou o arquivo.
- Um **grupo** proprietário. Normalmente é o grupo primário de quem criou o arquivo, mas este pode mudá-lo para outro grupo do qual ele também faça parte.
- **Permissões de acesso** definidas para o **usuário**, o **grupo** e **outros** usuários (terceiros).

As permissões definidas para os **arquivos** são:

- **Leitura**, permitindo acesso ao conteúdo do arquivo.
- **Escrita**, permitindo modificar o conteúdo do arquivo.
- **Execução**, permitindo executar o arquivo (caso seja um executável ou script).

As permissões definidas para os **diretórios** são similares:

- **Leitura**, permitindo acesso ao conteúdo do diretório (listar os arquivos presentes).
- **Escrita**, permitindo modificar o conteúdo do diretório (criar ou apagar arquivos).
- **Execução**, permitindo entrar no diretório, ou atravessá-lo.

Pode-se afirmar que um arquivo é protegido contra leituras ou modificações por suas próprias permissões, e contra apagamentos ou renomeações pelas permissões do diretório onde ele se encontra.

Consultando permissões

As permissões de acesso a arquivos e diretórios podem ser consultadas através de uma listagem de diretório longa, usando o comando `ls -l` (ou seu alias `ll`). Uma listagem típica seria:

```
-rw-r--r--   1 maziero  users        4068 mar 26 21:09 02.html
drwx------   5 maziero  users        1024 set  5  1998 Desktop/
drwx------   4 maziero  users        1024 jan 26  1998 administ/
drwxr-xr-x   2 maziero  users        1024 set 13  1998 axhome/
drwx------   2 maziero  users        1024 set  7  1998 bin/
-rw-r-----   1 maziero  users        4956 mar 26 20:34 descricao.html
drwx------  11 maziero  users        1024 jan 14 10:52 diversos/
drwx------   2 maziero  users        1024 jan 26  1998 ensino/
drwx------   2 maziero  users        1024 jan 26  1998 extensao/
drwx------   3 maziero  users        1024 mar  8  1998 formacao/
drwx------   4 maziero  users       13312 fev 23 20:49 icons/
drwx------   2 maziero  users        1024 ago  5  1998 mail/
drwx------   2 maziero  users        1024 jul  3  1998 nsmail/
drwx------   2 maziero  users        1024 out 13 19:22 pesquisa/
drwx------   8 maziero  users        1024 nov 24  1997 public_html/
drwx------  10 maziero  users        1024 mar 25 21:28 raytrace/
drwx------   3 maziero  users        1024 set 28 23:03 sistema/
drwxr-xr-x   3 maziero  users        1024 mar 26 21:07 testes/
drwx------   5 maziero  users        1024 out 17  1997 tex/
-rw-------   1 maziero  users        9718 ago  2  1998 wood.gif
```

Vamos analisar melhor os caracteres das colunas iniciais da listagem de diretório apresentada acima.

As entradas de diretório em um sistema UNIX têm seu tipo indicado pelo **primeiro caractere** da listagem de diretório longa. Os tipos de entradas mais frequentes são:

- `-` : arquivo normal
- `d` : diretório
- `l` : link simbólico (atalho)
- `b` : dispositivo (mapeado em `/dev/`) orientado a blocos (como os discos rígidos)
- `c` : dispositivo (mapeado em `/dev/`) orientado a caracteres (como modems e portas seriais)
- `s` : socket mapeado em arquivo (para comunicação entre processos)
- `p` : FIFO ou *Named Pipe* (outro meio de comunicação entre processos)

Os demais caracteres representam os direitos de acesso do usuário (*user*), do grupo (*group*) e de terceiros (*others*), em grupos de três caracteres:

- `r` : permissão de leitura (*read*).
- `w` : permissão de escrita (*write*).
- `x` : permissão de execução (*eXecute*).
- `-` : indica que o respectivo direito está negado.
- `s` : bits SUID e SGID setados (veremos mais tarde).

Vejamos um exemplo:

```
-rw-r----- 1 maziero  users  4956 mar 26 20:34 descricao.html
```

A linha de listagem acima indica que:

- A entrada corresponde a um arquivo normal (o primeiro caractere é `-`).
- O proprietário do arquivo `descricao.html` é o usuário `maziero`.
- O proprietário possui direito de leitura e escrita sobre o arquivo, mas não de execução.
- O arquivo também pertence ao grupo `users`.
- O grupo possui apenas direito de leitura sobre o arquivo.
- Outros usuários (terceiros) não possuem nenhum direito de acesso ao arquivo.

O comando chmod

Este comando 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 arquivo(s)*

A definição das permissões pode ser feita de forma **simbólica** ou **octal**. A forma simbólica é a mais simples e por isso a mais usada por debutantes. A forma octal é no entanto mais empregada, sobretudo em scripts antigos. Neste texto vamos nos restringir à forma simbólica. 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

Finalmente, as letras do terceiro grupo indicam que permissões devem ser alteradas:

- r : permissão de leitura
- w : permissão de escrita
- x : permissão de execução (ou acesso ao diretório)
- X : permissão de execução, caso algum membro (u, g, o) já a possua
- u : usar as permissões já atribuídas ao usuário proprietário
- g : usar as permissões já atribuídas ao grupo proprietário
- o : usar as permissões já atribuídas a outros

Vejamos alguns exemplos:

- `chmod o-w *.c` : retira de terceiros a permissão de escrita sobre todos os arquivos C no diretório corrente.

```
$ ls -l
-rw-rw-rw-  1 maziero    prof   523   Mar 27   08:51 main.c
-rwxrwxrwx  1 maziero    prof  2321   Mar 25   09:37 funct.c

$ chmod o-w *.c

$ ls -l
-rw-rw-r--  1 maziero    prof   523   Mar 27   08:51 main.c
-rwxrwxr-x  1 maziero    prof  2321   Mar 25   09:37 funct.c
```

- `chmod go-rwx ~/*` : retira do grupo e de terceiros todas as permissões (leitura, escrita, execução) sobre todos os arquivos do diretório home.

```
$ ls -l
-rw-rw-rw-  1 maziero    prof   523   Mar 27   08:51 main.c
-rwxrwxrwx  1 maziero    prof  2321   Mar 25   09:37 funct.c
-rw-r--r--  1 maziero    prof 75643   Mar 27   08:56 main.o

$ chmod go-rwx ~/*

$ ls -l
-rw-------  1 maziero    prof   523   Mar 27   08:51 main.c
-rwx------  1 maziero    prof  2321   Mar 25   09:37 funct.c
-rw-------  1 maziero    prof 75643   Mar 27   08:56 main.o
```

- `chmod u+w,go=r *.txt` : dá 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:

```
$ ls -l
-r--rw-rw-  1 maziero    prof  2386   Mar 27   08:51 readme.txt
-rwxrwxrw-  1 maziero    prof 12875   Mar 25   09:37 instal.txt

$ chmod u+w,go=r *.txt
```

```
$ ls -l
-rw-r--r--  1 maziero    prof  2386   Mar 27   08:51 readme.txt
-rwxr--r--  1 maziero    prof 12875   Mar 25   09:37 instal.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  ~
```

O uso do comando `chmod` em modo octal é similar ao modo simbólico, embora mais difícil. 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 considerar que `rwxr-x---` -> `rwx r-x ---` -> 111 101 000 (binário) -> 7 5 0 (octal) -> 750. Desta forma, o comando a executar é:

```
chmod 750 teste.c
```

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 Unix, mesmo os mais antigos. Além disso, sua compreensão é importante para o uso do comando `umask`.

O comando umask

O comando `umask` permite definir uma *máscara padrão de permissões* para a criação de novos arquivos e diretórios. A sintaxe desse comando usa a notação octal, para definir as permissões a suprimir nos novos arquivos e diretórios, a partir das permissões máximas. Vejamos um exemplo:

> `r w x r - x - - -` -> permissões desejadas para os novos arquivos
> `- - - - w - r w x` -> permissões a suprimir
> `0 0 0 0 1 0 1 1 1` -> permissões a suprimir, em binário (000, 010 e 111)
> `0 2 7` -> máscara de permissões, em octal

Assim, o comando `umask 027` permite definir a máscara desejada (`rwxr-x---`). Normalmente esse comando é usado nos arquivos de configuração do shell, e nos scripts de instalação de aplicações.

Os comandos chown e chgrp

O comando `chown` permite a mudança do usuário proprietário de um arquivo. Somente o superusuário pode fazê-lo. O comando `chgrp` permite a permite a mudança do grupo proprietário de um arquivo. Somente o superusuário (*root*) e o usuário proprietário do arquivo podem fazê-lo. O proprietário só pode mudá-lo para um grupo ao qual ele também pertença.

Exemplos:

```
$ ls -l
drw-------  2 maziero    prof   0   Mar 27   08:51 dir1

$ chown joao dir1

$ ls -l
drw-------  2 joao       prof   0   Mar 27   08:51 dir1
```

```
$ chgrp labin dir1

$ ls -l
drw-------  2 joao       labin  0   Mar 27   08:51 dir1
```

O comando newgrp

Permite ao usuário mudar seu grupo principal para outro grupo ao qual ele também pertença. Todos os arquivos e diretórios criados a partir dessa mudança pertencerão ao novo grupo, e não mais ao grupo primário do usuário. Por exemplo:

```
$ mkdir dir1

$ ls -l
drw-------  2 maziero    prof   0   Mar 27   08:51 dir1

$ newgrp larsis

$ mkdir dir2

$ ls -l
drw-------  2 maziero    prof   0   Mar 27   08:51 dir1
drw-------  2 maziero    larsis 0   Mar 27   08:52 dir2
```

Para voltar ao grupo primário basta executar `exit`. Para saber a quais os grupos primário e secundários do usuário basta executar o comando `id`.

Exercícios

1. Crie a seguinte estrutura de diretórios na sua area HOME:

   ```
   ~/infraComp/aula01
          |__ /aula02
                   |__ /exercicios
          |__ /aula03
          |__ /aula04
          |__ /aula05
   ```

2. Em cada diretório acima crie um arquivo chamado DATA, utilizando o comando `touch`.
3. Altere as permissões de acesso de seu diretório HOME para que somente você e os usuários do seu grupo tenham acesso de leitura e execução, e você tenha acesso de leitura, escrita e execução.
4. Crie um diretório `$HOME/mydir` e ajuste suas permissões de forma que seus colegas consigam criar um arquivo nele. Crie um arquivo com seu nome no diretório correspondente do seu vizinho.
5. Qual a diferença no resultado da execução dos dois comandos abaixo?

   ```
   cp teste teste.bak
   mv teste teste.bak
   ```

6. Qual a diferença entre executar e ler um diretório (permissões r e x)?
7. Qual a diferença entre executar e ler um arquivo (permissões r e x)?
8. Você conseguiria mudar o nome de seu próprio diretório *home*? Por que?
9. Crie um arquivo `teste` com os direitos de acesso `rw-rw-rw-`, e indique como usar o comando `chmod` para alterar seus direitos de acesso para:

    - `rw-rw-r--`
    - `r-xr-xr-x`
    - `rw-r--r--`
    - `r--------`
10. Execute o comando `umask` para que novos arquivos criados no sistema tenham as permissões de acesso definidas a seguir, e teste as máscaras definidas criando novos arquivos (comando `touch`) e diretórios (comando `mkdir`). Finalmente, explique por que razão as permissões dos arquivos não coincidem com as esperadas, mas as dos diretórios sim.
    - `rw-rw-r--`
    - `r-xr-xr-x`
    - `rw-r--r--`
    - `r--------`
11. Crie dois diretórios `d1` e `d2`, com permissões respectivas `r--r--r--` e `r-xr-xr-x`, e compare as possibilidades de acesso em ambos. É possível listar o conteúdo de ambos, estando fora deles? É possível entrar em ambos?

From:
https://wiki.inf.ufpr.br/maziero/ - **Prof. Carlos Maziero**

Permanent link:
https://wiki.inf.ufpr.br/maziero/doku.php?id=unix:permissoes_em_arquivos

Last update: **2023/11/16 17:33**