Permissão de Acesso a Arquivos e Diretórios

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro
Prof. Edwar Saliba Júnior
Abril / 2025

Donos, Grupos e Outros Usuários

- As permissões de acesso protegem o sistema de arquivos GNU/Linux de acessos indevidos de pessoas ou programas não autorizados;

- as permissões de acesso do GNU/Linux também impede que um programa mal intencionado, por exemplo, apague um arquivo que não deve, envie arquivos especiais para outras pessoas ou forneça acesso da rede para que outros usuários invadam o sistema e

- a ideia básica da segurança no sistema GNU/Linux é definir o acesso aos arquivos por donos, grupos e outros usuários.

Dono (owner)

- **dono**
  - é a pessoa que criou o arquivo ou o diretório. O nome do dono do arquivo/diretório é o mesmo do usuário 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 usuário ao qual o arquivo pertence é armazenada no arquivo **`/etc/passwd`** e do grupo no arquivo **`/etc/group`**. Estes são arquivos textos comuns e podem ser editados em qualquer editor de texto.

Grupo (group)

- **`grupo`**
  - permite que vários usuários diferentes tenham acesso a um mesmo arquivo. Cada usuário pode fazer parte de um ou mais grupos e então acessar arquivos que pertençam ao mesmo grupo que o seu;
  - por padrão, quando um novo usuário é criado e se não foi especificado nenhum grupo, ele pertencerá ao grupo de mesmo nome do seu grupo primário (este comportamento é controlado pelo parâmetro `USERGROUPS=yes` do arquivo `/etc/adduser.conf`). A identificação do grupo é chamada de `GID` (`group id`) e
  - um usuário pode pertencer a um ou mais grupos.

Outros Usuários (others)

- **outros**
  - é a categoria de usuários que não são donos e não pertencem ao grupo do arquivo;
  - cada um dos tipos acima possuem três tipos básicos de permissões de acesso: **r**, **w** e **x**; as quais detalharemos a seguir.

Tipos de Permissões

- Os tipos de permissões que se aplicam ao dono, grupo e outros usuários, são três:

  `r` (read) - permissão de leitura para arquivos. Caso seja um diretório, permite listar seu conteúdo;

  `w` (write) - permissão de gravação para arquivos. Caso seja um diretório, permite a gravação de arquivos ou outros diretórios dentro dele. Para que um arquivo/diretório possa ser criado ou apagado, é necessário a permissão de gravação e

  `x` (execute) - permissão para executar um arquivo. Caso seja um diretório, permite que seja acessado através do comando `cd`.

- As permissões de acesso a um arquivo/diretório podem ser visualizadas com o uso do comando `ls -l`.

- As 3 letras (`rwx`) são agrupadas da seguinte forma:

```
-rw-r--r-- 1 edwar edwar   115 Apr 13 16:05 agenda.txt
drwxrw-rw- 2 edwar edwar  4096 Mar 30 19:40 Arquivos
           (dono) (grupo)
```

Tipos de Permissões de Acesso

- As 3 letras (**rwx**) são agrupadas da seguinte forma:

```
-rwxr-xr-- 1 gleydson users 8192 nov 4 16:00 teste
drwxrwxrwx 2 gleydson users 1022 nov 4 17:55 exemplo
```

dono grupo outros — dono — grupo

  - o primeiro caractere, em vermelho, diz qual é o tipo do arquivo. Se for:
    - "d" é um diretório,
    - "l" é um *link* e
    - "-" é um arquivo;
  - do segundo ao quarto caractere (rwx) são as permissões de acesso do dono do arquivo. Neste caso o gleydson, ele tem a permissão de ler, gravar e executar o arquivo teste;
  - do quinto ao sétima caractere (r-x) são as permissões de acesso do grupo do arquivo. Neste caso todos os usuários que pertencem ao grupo users têm permissão de ler e também executar o arquivo teste e
  - do oitavo ao décimo caractere (r--) são as permissões de acesso para os outros usuários. Neste caso todos os usuários que não são donos do arquivo teste e que não estão no grupo users têm somente a permissão de ler o arquivo teste;

Tipos de Permissões de Acesso

- observações:
  - o usuário `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.

A Conta `root`

- A conta `root` é também chamada de superusuário, este é um login que não possui restrições de segurança. A conta `root` somente deve ser usada para fazer a administração do sistema e o menor tempo possível;

- recomenda-se que qualquer senha deve conter, no mínimo, de 6 a 8 caracteres entre: letras maiúsculas e minúsculas, caracteres especiais, números e etc.

Comando `chmod`

- Este comando serve para mudar as permissões de acesso de um arquivo ou diretório.
- Sempre que um arquivo é criado, seu dono é o usuário que o criou e seu grupo é o grupo do usuário.

```
chmod [opções] [permissões] [diretório/arquivo]
```

- onde:

`diretório/arquivo`
é o diretório ou arquivo que terá sua permissão alterada;

`opções`

`-v, --verbose`
mostra todos os arquivos que estão sendo processados;

`-f, --silent`
não mostra a maior parte das mensagens de erro.

Comando chmod

- onde:
  - `-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 subdiretórios;
  - `ugoa+-=rwxXst`
    `ugoa` - controla que nível de acesso será mudado. Especificam, em ordem, `dono` (**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`;

- 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 usuários` (**o**) serão idênticas ao do `dono` (**u**). Então a nova permissão de acesso do arquivo `teste.txt` será `r-xr--r-x`

Comando `chmod`

- Exemplos de permissões de acesso:

```
chmod g+r *
```

  - permite que todos os usuários que pertençam ao grupo dos arquivos tenham permissão de leitura em todos os arquivos do diretório atual;

```
chmod o-r teste.txt
```

  - retira a permissão de leitura do arquivo `teste.txt` para os outros usuários do arquivo `teste.txt`;

```
chmod uo+x teste.txt
```

  - inclui a permissão de execução do arquivo `teste.txt` para o dono e outros usuários do arquivo;

```
chmod a+x teste.txt
```

  - inclui a permissão de execução do arquivo `teste.txt` para o dono, grupo e outros usuários;

```
chmod a=rw teste.txt
```

  - define a permissão de todos os usuários para leitura e gravação do arquivo `teste.txt`.

Comando `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/subdiretórios` do diretório atual.

- Para saber em quais grupos um usuário do sistema operacional está, basta executar os comandos:
  - `groups` (para o usuário corrente) ou
  - `groups + "nome do usuário"`

Comando `chown`

- Muda o 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 e
    - `-f, --supress`
      não mostra mensagens de erro durante a execução do programa.

Comando `chown`

- onde:
  - `-c, --changes`
    mostra somente arquivos que forem alterados;
  - `-R, --recursive`
    altera dono e grupo de arquivos no diretório atual e subdiretórios.

- 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 gleydson teste.txt` - muda o dono do arquivo `teste.txt` para `gleydson`;

  `chown gleydson.foca teste.txt` - muda o dono do arquivo `teste.txt` para `gleydson` e seu grupo para `foca` e

  `chown -R gleydson.focalinux *` - muda o `dono/grupo` dos arquivos do diretório atual e subdiretórios para `gleydson/focalinux` (desde que você tenha permissões de gravação no diretórios e subdiretórios).

Modo de Permissão Octal

- Ao invés de se utilizar os modos de permissão `+r, -r` e etc., pode-se usar 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. Gerenciar as permissões de acesso usando o modo octal pode ser mais fácil. Pois, você especifica a permissão do `dono`, `grupo` e `outros usuários` com apenas 3 números. A seguir a lista de permissões de acesso octal:

Binário	Octal	Simbólico
000	0	---
001	1	--x
010	2	-w-
011	3	-wx
100	4	r--
101	5	r-x
110	6	rw-
111	7	rwx

Modo de Permissão Octal

- O uso de um destes números define a permissão de acesso do `dono`, `grupo` ou `outros usuários`.

```
1 = Executar
2 = Gravar
4 = Ler
```

- Basta agora fazer o seguinte:
  permissão só de execução, use **1**
  permissão só de leitura, use **4**
  permissão só de gravação, use **2**
  permissão de leitura e gravação, use **6** (equivale a **4+2**)
  permissão de leitura e execução, use **5** (equivale a **4+1**)
  permissão de execução e gravação, use **3** (equivale a **1+2**) e
  permissão de leitura, gravação e execução, use **7** (equivale a **4+2+1**).

Modo de Permissão Octal

- Alguns exemplos:

```
chmod 764 teste
```

- a permissão de acesso 764 significa, dono (7), grupo (6) e outros usuários (4). Ou seja:
  - **dono (7)** tem permissão de **leitura, gravação e execução (rwx)** ao arquivo **teste**,
  - **grupo** (6) tem permissão de **leitura** e **gravação** (**rw**) e
  - **outros usuários** (4) têm acesso somente **leitura** (**r**) ao arquivo **teste**.

- Outro exemplo:

```
chmod 40 teste
```

- o exemplo acima define a permissão de acesso dos **outros usuários** (**0**) como zero ou nenhuma permissão. E define a permissão de acesso do **grupo** (4) como somente **leitura** (**r**). Note que foi usado somente dois números e então a permissão de acesso do dono do arquivo não é modificada. Pois, as permissões de acesso são interpretadas, pelo computador, da direita para a esquerda!

```
chmod 751 teste
```

- O exemplo acima define a permissão de acesso dos **outros usuários** (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**).

Comando `umask`

- A `umask` (**`user mask`**) são **4** números que definem as permissões que serão mascaradas (removidas) das **próprias permissões do `umask`** do **dono**, do **grupo** e de **outros usuários** que o **arquivo/diretório** receberá quando for criado ou copiado para um novo local. Digite `umask` sem parâmetros para retornar o valor da `umask` de seu usuário atual no sistema;

- A `umask` tem efeitos diferentes caso o arquivo que estiver sendo criado for:
  - binário ou
  - texto ou
  - diretório, vejamos no próximo *slide*.

Comando umask

- Veja a tabela a seguir para ver qual é a mais adequada a sua situação:

UMASK	ARQUIVO		DIRETÓRIO
	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	---	---	---

- ao executar o comando `umask` é muito comum ter como resultado o número 0022.

Bit especial

Comando `umask`

- Um arquivo texto criado com o comando **`umask 0012;`**
  - **`touch texto.txt`** receberá as permissões **`-rw-rw-r--`**, pois **`0`** (**dono**) terá permissões **`rw-`**, **`1`** (**grupo**), terá permissões **`rw-`** e **`2`** (**outros usuários**) 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 no *slide* anterior).
- Por este motivo é preciso 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 **`0022`**. A **`umask`** padrão no sistema Debian é a **`0022`**;
- A **`umask`** é de grande utilidade para programas que criam **`arquivos/diretórios`** temporários, desta forma pode-se bloquear o acesso de outros usuários desde a criação do arquivo, evitando recorrer ao **`chmod`**.

Referências

- GUIA FOCA GNU/Linux. Iniciante. Disponível em: <http://www.guiafoca.org/cgs/guia/iniciante/ch-perm.html>. Acesso em: 06 ago. 2017.