

Linux Permissões de arquivos com chmod

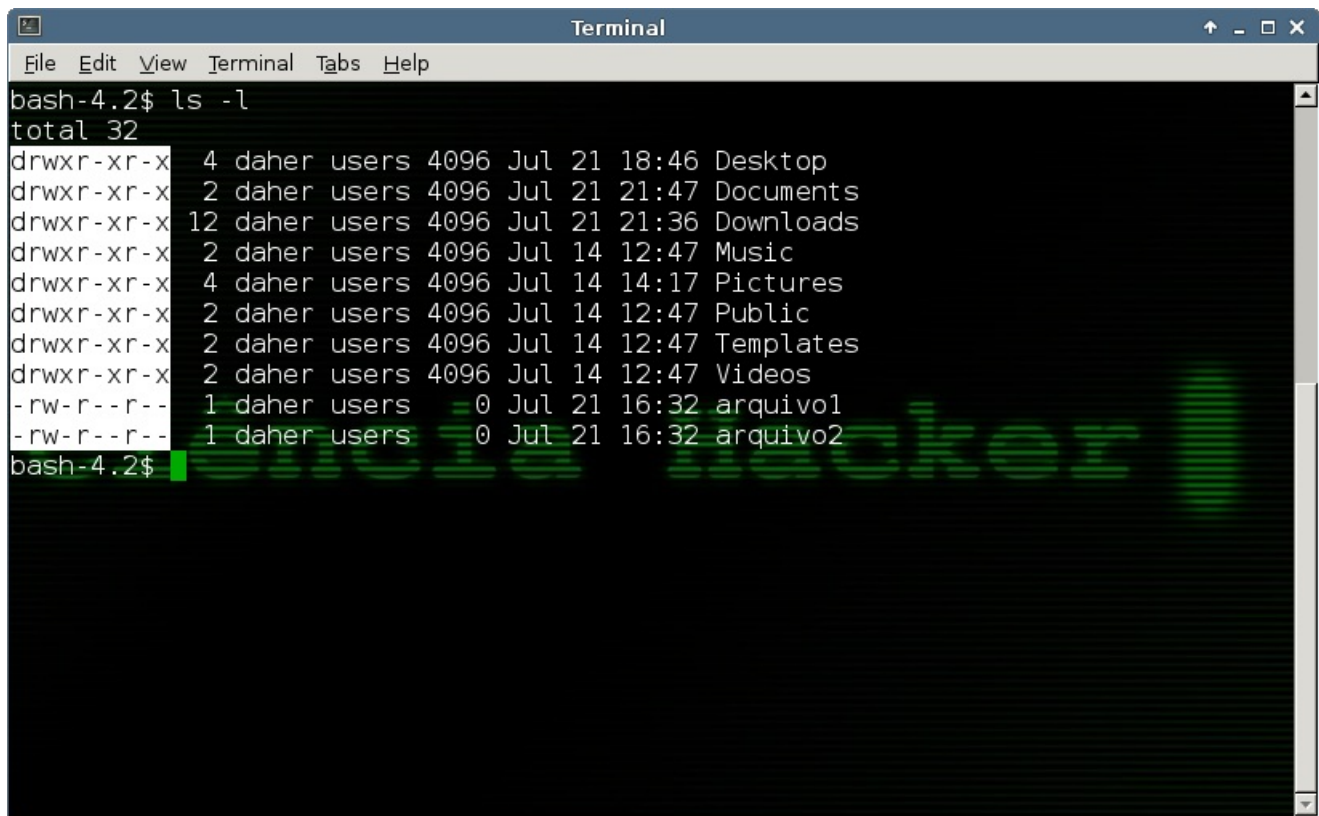
[23 de julho de 2015](#) / [daher](#)

Algo muito discutido e até “amedrontador” para alguns usuários Linux é a questão de permissões de arquivos. Muitos usuários veem esse assunto como uma verdadeira “hidra”. Então, caro usuário, aproxime-se e elimine de vez esse pensamento de sua cabeça.

Neste tópico aprenderemos a utilizar o comando `chmod` para alterar as permissões de arquivos e diretórios do sistema. Esse assunto é muito abordado em questões de provas, como as famosas **LPIs**.

O comando `ls`

O comando `ls` seguido da opção `-l` lista os arquivos de determinado diretório e exibe algumas informações sobre eles, inclusive suas permissões. Na imagem abaixo, nosso diretório corrente é o **/home/** e estamos exibindo as permissões dos arquivos e diretórios no quadro destacado:



```
bash-4.2$ ls -l
total 32
drwxr-xr-x  4 daher users 4096 Jul 21 18:46 Desktop
drwxr-xr-x  2 daher users 4096 Jul 21 21:47 Documents
drwxr-xr-x 12 daher users 4096 Jul 21 21:36 Downloads
drwxr-xr-x  2 daher users 4096 Jul 14 12:47 Music
drwxr-xr-x  4 daher users 4096 Jul 14 14:17 Pictures
drwxr-xr-x  2 daher users 4096 Jul 14 12:47 Public
drwxr-xr-x  2 daher users 4096 Jul 14 12:47 Templates
drwxr-xr-x  2 daher users 4096 Jul 14 12:47 Videos
-rw-r--r--  1 daher users   0 Jul 21 16:32 arquivo1
-rw-r--r--  1 daher users   0 Jul 21 16:32 arquivo2
bash-4.2$
```

Na imagem acima, podemos observar diversos caracteres. O primeiro caractere indica o tipo do arquivo. Na imagem, observamos a presença do caractere **d**, que indica **diretório**. Existem ao todo cinco caracteres que indicam o tipo do arquivo. São eles:

- (-) = Arquivo comum de usuário
- (b) = Arquivo de bloco
- (c) = Arquivo de caractere
- (d) = Diretório
- (l) = Link

A partir da segunda coluna, encontramos as permissões de cada arquivo. Podemos observar a presença dos caracteres **-**, **r**, **w** e **x**. Abaixo, encontram-se o significado de cada um deles:

- (-) = Nenhuma permissão
- (r) = Read (Leitura)
- (w) = Write (Escrita)
- (x) = Execution (Execução)

Os tipos de usuários

Agora, você deve estar questionando-se sobre a presença de vários caracteres (citados no último tópico) em apenas um arquivo. Isso acontece porque devemos alterar as permissões para as três

entidades do sistema: **usuário**, **grupo** e **outros**.

O **usuário**, no caso, será o criador daquele arquivo. Consequentemente, *o usuário pertence a um grupo no sistema*. Se o **grupo** possuir a permissão de escrita, por exemplo, todos os usuários que pertencem a esse grupo terão permissões de escrita. Os **outros** são nada mais, nada menos que usuários que estão fora do grupo (como aplicativos, usuários de terceiros, etc.). No arquivo **arquivo1**, por exemplo, o **usuário** tem as permissões de **leitura** e **escrita**; o **grupo** tem permissão apenas de **leitura** e os **outros** também possuem apenas permissão de **leitura**.

O comando *chmod*

O comando `chmod` é utilizado para alterar permissões de arquivos e/ou diretórios. Sua sintaxe básica é:

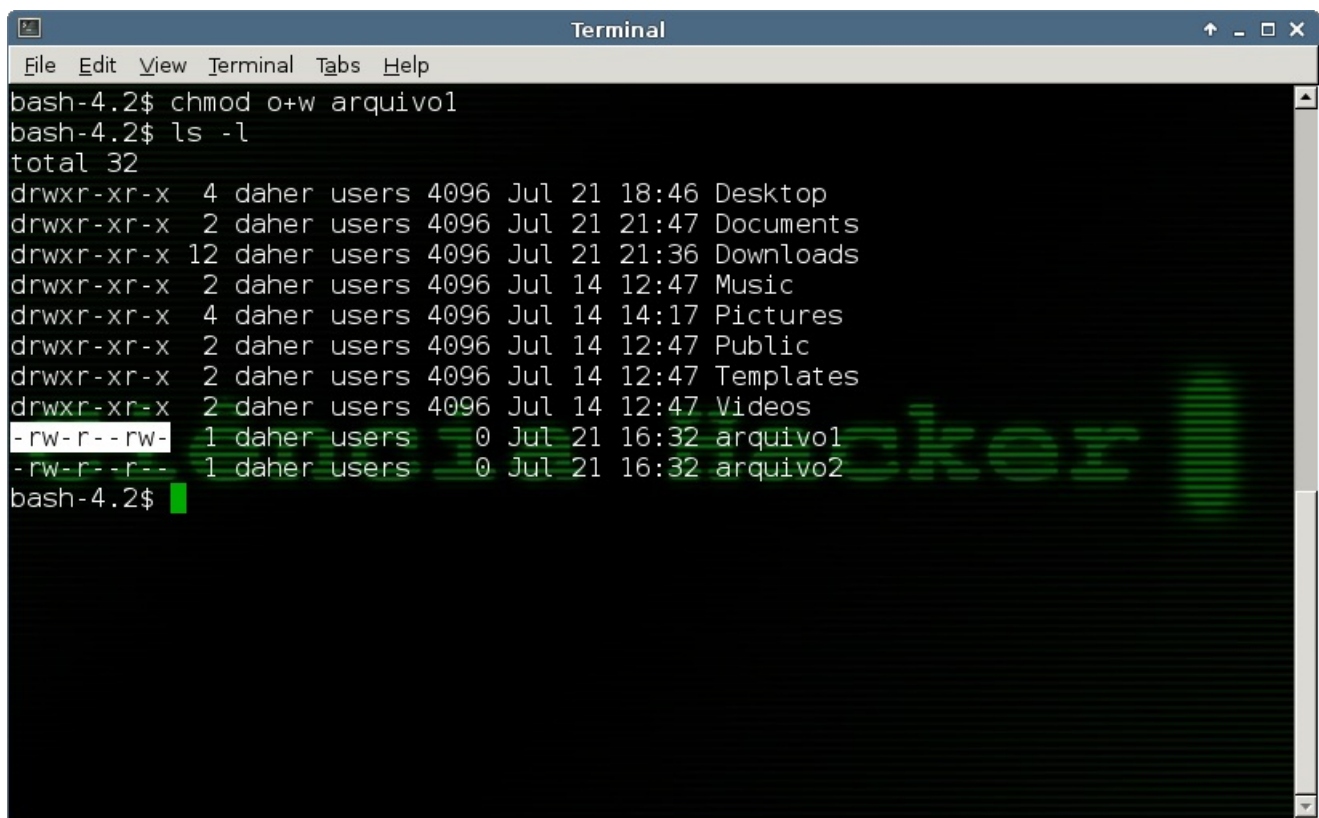
```
chmod permissão arquivo_ou_diretório
```

Alterando as permissões com *chmod*

Existem duas formas de se alterar as permissões nos arquivos e diretórios com o `chmod`. A primeira forma é especificando o usuário e as permissões que ele terá no arquivo. Por exemplo, para adicionarmos a permissão de **escrita** no arquivo **arquivo1**, utilizaremos um `chmod`, a letra que simboliza os **outros** usuários, que no caso é a letra **o**, o sinal de adição (+), a permissão desejada e o nome do arquivo. Ficaria exatamente deste modo:

```
chmod o+w arquivo1
```

Como podemos observar na imagem abaixo, a permissão foi alterada:



```
Terminal
File Edit View Terminal Tabs Help
bash-4.2$ chmod o+w arquivo1
bash-4.2$ ls -l
total 32
drwxr-xr-x  4 daher users 4096 Jul 21 18:46 Desktop
drwxr-xr-x  2 daher users 4096 Jul 21 21:47 Documents
drwxr-xr-x 12 daher users 4096 Jul 21 21:36 Downloads
drwxr-xr-x  2 daher users 4096 Jul 14 12:47 Music
drwxr-xr-x  4 daher users 4096 Jul 14 14:17 Pictures
drwxr-xr-x  2 daher users 4096 Jul 14 12:47 Public
drwxr-xr-x  2 daher users 4096 Jul 14 12:47 Templates
drwxr-xr-x  2 daher users 4096 Jul 14 12:47 Videos
-rw-r--rw-  1 daher users   0 Jul 21 16:32 arquivo1
-rw-r--r--  1 daher users   0 Jul 21 16:32 arquivo2
bash-4.2$
```

Para removermos a permissão de **escrita** dos **outros** usuários, bastaria trocar o caractere de adição (+) por um caractere de subtração (-).

```
chmod o-w arquivo1
```

Como podemos observar abaixo, as permissões foram alteradas mais uma vez:

```
Terminal
File Edit View Terminal Tabs Help
bash-4.2$ chmod o-w arquivo1
bash-4.2$ ls -l
total 192
drwxr-xr-x  4 daher users   4096 Jul 21 18:46 Desktop
drwxr-xr-x  2 daher users   4096 Jul 21 21:47 Documents
drwxr-xr-x 12 daher users   4096 Jul 21 21:36 Downloads
drwxr-xr-x  2 daher users   4096 Jul 14 12:47 Music
drwxr-xr-x  4 daher users   4096 Jul 14 14:17 Pictures
drwxr-xr-x  2 daher users   4096 Jul 14 12:47 Public
drwxr-xr-x  2 daher users   4096 Jul 14 12:47 Templates
drwxr-xr-x  2 daher users   4096 Jul 14 12:47 Videos
-rw-r--r--  1 daher users     0 Jul 21 16:32 arquivo1
-rw-r--r--  1 daher users     0 Jul 21 16:32 arquivo2
-rw-r--r--  1 daher users 160438 Jul 21 21:48 chmod o+w.png
bash-4.2$
```

Há também um outro meio de realizar a alteração das permissões: por meio da especificação das permissões em números. Este meio é o “bicho de sete cabeças” para alguns usuários.

Muitos usuários ficam aterrorizados ao depararem com a tabela de permissões abaixo:

| Valor | Permissão |
|-------|------------------------------|
| 0 | Nenhuma |
| 1 | Execução |
| 2 | Escrita |
| 3 | Execução + Escrita |
| 4 | Leitura |
| 5 | Execução + Leitura |
| 6 | Leitura + Escrita |
| 7 | Execução + Escrita + Leitura |

São também muitos os que não sabem que o “macete” da tabela é decorar apenas o número das permissões de **execução**, **escrita** e **leitura**. Sabendo essas três, as restantes são a soma. Por exemplo, se quisermos saber o número da permissão de **leitura + escrita**, basta somarmos os valores dessas duas opções; ou seja, $4 + 2 = 6$. Sendo assim, a opção 6 será a opção de **leitura + escrita**.

Sabendo essa regra, basta agora sabermos a ordem dos usuários. Devemos colocar em mente a seguinte ordem: **usuário** -> **grupo** -> **outros**. Logo, se quisermos atribuir as seguintes permissões para os usuários:

- Usuário: escrita + leitura
- Grupo: leitura
- Outros: nenhuma

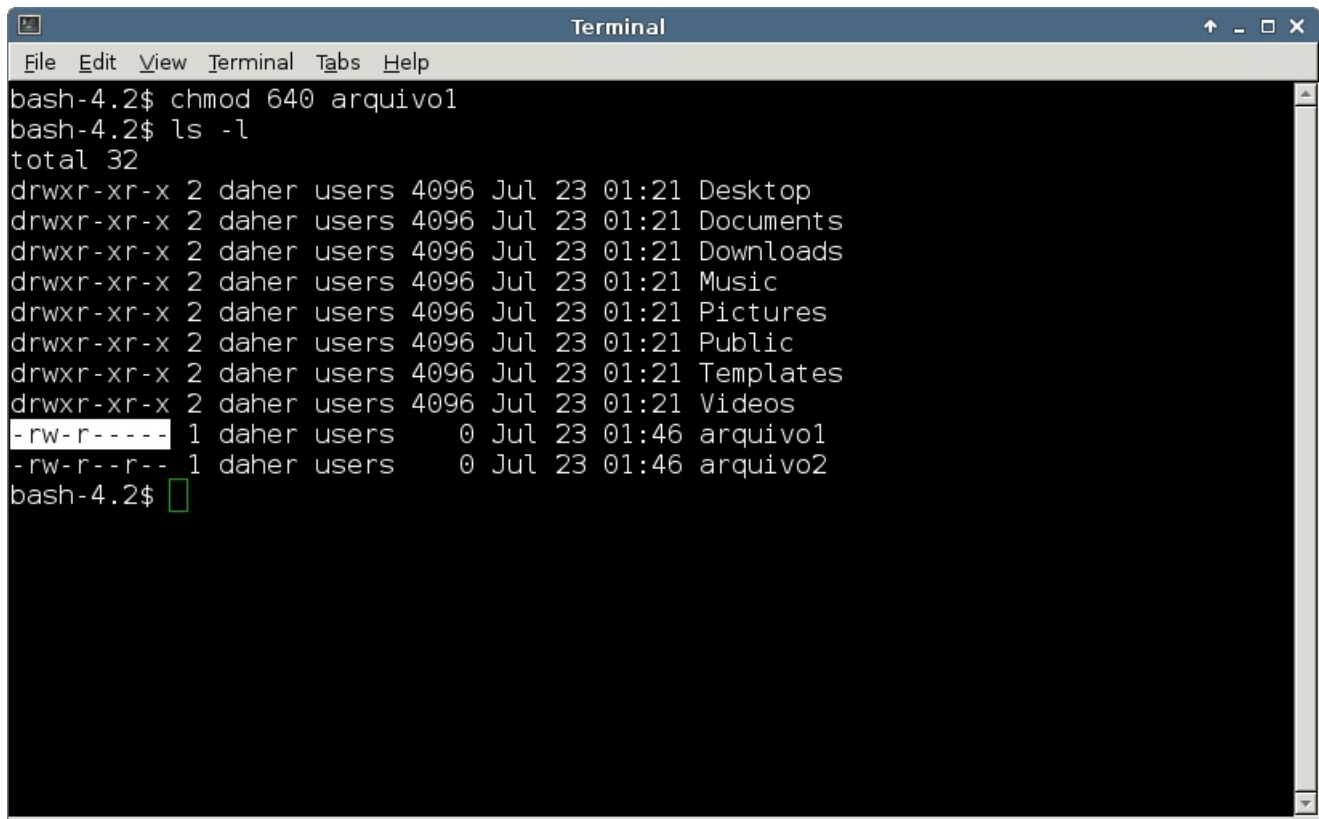
Deveremos realizar o somatório de cada uma das permissões:

- Usuário: escrita + leitura -> $2 + 4 = 6$
- Grupo: leitura -> 4
- Outros: nenhuma -> 0

Realizada a soma, poderemos atribuir as permissões com o comando `chmod`. O comando, então, ficará da seguinte maneira (lembrando que deve-se seguir a ordem apresentada acima - usuário -> grupo -> outros):

```
chmod 640 arquivo1
```

Feito isso, cada usuário terá as devidas permissões sobre o arquivo **arquivo1**, como apresentado abaixo:

A screenshot of a Linux terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal shows the following commands and output:

```
bash-4.2$ chmod 640 arquivo1
bash-4.2$ ls -l
total 32
drwxr-xr-x 2 daher users 4096 Jul 23 01:21 Desktop
drwxr-xr-x 2 daher users 4096 Jul 23 01:21 Documents
drwxr-xr-x 2 daher users 4096 Jul 23 01:21 Downloads
drwxr-xr-x 2 daher users 4096 Jul 23 01:21 Music
drwxr-xr-x 2 daher users 4096 Jul 23 01:21 Pictures
drwxr-xr-x 2 daher users 4096 Jul 23 01:21 Public
drwxr-xr-x 2 daher users 4096 Jul 23 01:21 Templates
drwxr-xr-x 2 daher users 4096 Jul 23 01:21 Videos
-rw-r----- 1 daher users    0 Jul 23 01:46 arquivo1
-rw-r--r-- 1 daher users    0 Jul 23 01:46 arquivo2
bash-4.2$
```

The permissions for `arquivo1` are highlighted with a white box, showing `-rw-r-----`.

Como observamos, o usuário criador do arquivo ficou com permissão de escrita e leitura, o grupo com apenas leitura e os demais usuários não ficaram com nenhuma permissão.

, ,