Permissões de Arquivos e Directórios

Administação de Sistemas Linux

Introdução

- Sistemas operacionais multiusuário têm de zelar pela segurança e privacidade dos dados individuais de cada usuário, bem como prezar pela integridade do sistema. Para isso existe as chamadas permissões de acesso, que actuam em dois aspectos fundamentais, o primeiro é a **privacidade** e o segundo, a **segurança.**
- Quanto à privacidade, as permissões permitem que os usuários restrinjam o acesso a seus arquivos, caso esses possuam algum conteúdo confidencial ou algo parecido. O dono de um arquivo, pode especificar quem pode ler o seu arquivo, quem pode modificá-lo, apagá-lo ou executá-lo, tratando-se de um executável.
- Quanto à segurança, um sistema nem sempre será utilizado somente por pessoas que possuem um bom conhecimento operacional. Sendo assim, estarão sujeitas a cometer erros diversos, desde os mais banais até erros graves, como excluir arquivos essenciais para o funcionamento de alguns programas ou do sistema todo.

Introdução (cont.)

- Assim, as permissões podem seguramente agir impedindo que determinados usuários possam excluir arquivos importantes ou mesmo executar programas que possam causar problemas ao sistema.
- O GNU/Linux tem um método muito simples de lidar com permissões que consiste em implementar controlos de acesso utilizando algumas propriedades dos ficheiros/directórios.
- As permissões de acesso a um arquivo/diretório podem ser visualizadas com o uso do comando ***ls -l.***

```
carlos@ubserver:~$ ls -l
total 8
-rw-r--r-- 1 carlos carlos  110 Nov 23 15:32 teste1
-rw-rw-r-- 1 carlos carlos    0 Nov 23 14:58 teste2
drwxrwxr-x 2 carlos carlos 4096 Nov 23 14:56 trabalhos
```

Introdução (cont.)

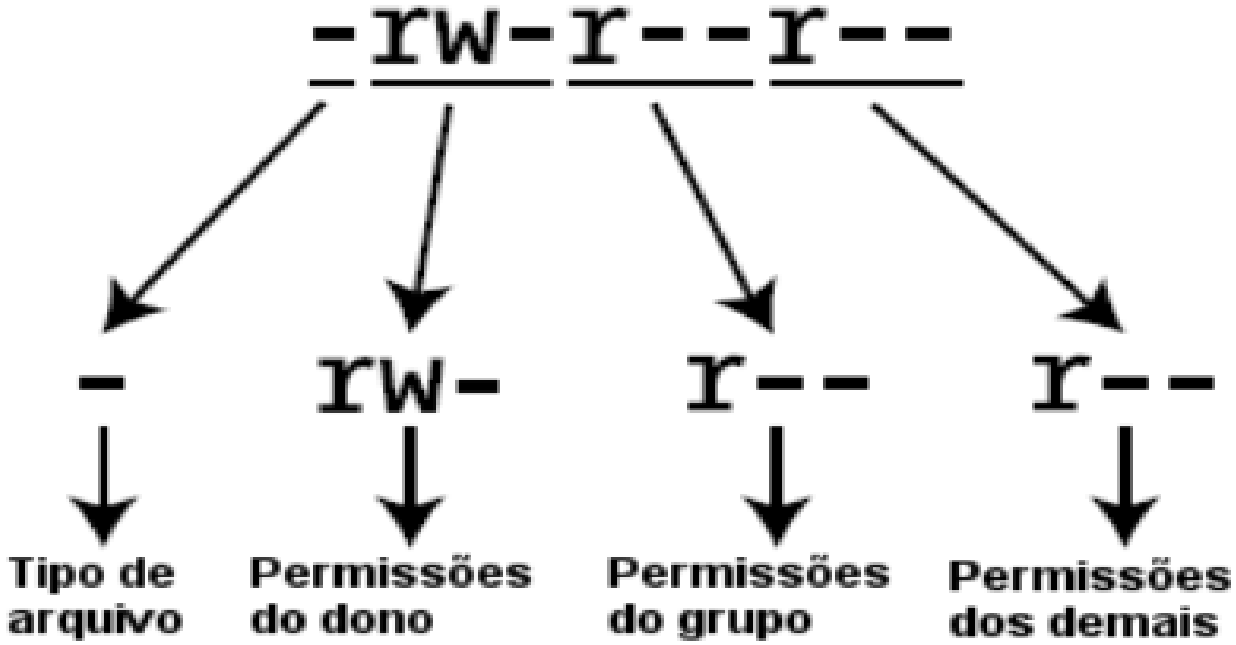
-rw-r--r-- 1 carlos carlos 110 Nov 23 07:18 trabalhos

1 2 3 4 5 6 7

1	**~rw~r~~r~~**	Permissões de acesso do arquivo/directório
2	1	Número de *links* do arquivo ou diretório
3	carlos	Dono
4	carlos	Grupo
5	110	Tamanho
6	Nov 23 07:18	Data/Hora da Modificação
7	trabalhos	Nome do arquivo/directório

Permissões de acesso de arquivos

O Linux trata **todos os diretorios como arquivos**, portanto, as permissoes se aplicam de igual forma para ambos. Tais permissoes podem ser divididas em quatro partes para indicar: tipo, proprietario, grupo e outras permissoes.



Permissões de acesso

Tipos de arquivos

A primeira letra da saída de ls -l descreve o tipo de arquivo. Os tipos de arquivo mais comuns são:

- **- (arquivo normal)** - Um arquivo pode conter qualquer tipo de dado. os arquivos podem ser modificados, movidos, copiados e excluídos.
- **d (diretório)** - Um diretório contém outros arquivos e diretórios e ajuda a organizar o sistema de arquivos. Tecnicamente, os diretórios são um tipo especial de arquivo.
- **l (link simbólico)** - Este "arquivo" é um apontador para outro arquivo ou diretório em outro local do sistema de arquivos.
- **b (dispositivo de bloco)** - Este arquivo representa um dispositivo virtual ou físico, como os discos ou outro tipo de dispositivo de armazenamento. Por exemplo, o primeiro disco rígido do sistema pode ser representado por /dev/sda.

Tipos de arquivos (cont.)

- **c (dispositivo de caracteres)** - Este arquivo representa um dispositivo virtual ou físico. Os terminais (como o terminal principal em /dev/tty) e portas seriais são exemplos comuns de dispositivos de caracteres.
- **s (socket)** - Os sockets são como “pipes” que transmitem informações entre dois programas.

- Não altere nenhuma das permissões dos dispositivos de bloco, de caracteres ou *sockets*, porque isso pode afectar o funcionamento do sistema,

Permissões de acesso

Donos, Grupos e Outros Usuários

- O princípio da segurança no sistema de arquivos GNU/ Linux é definir o acesso aos arquivos por donos, grupos e outros usuários
  - **Dono**: Chamamos de dono o usuário que criou o arquivo. O sistema de permissões no GNU/Linux permite que alteremos as permissões para nós próprios. Podemos, assim, evitar que, por exemplo, façamos alterações acidentais em arquivos importantes.
  - **Grupo**: Como sabemos, todo usuário do sistema GNU/Linux pertence a pelo menos um grupo. Assim, podemos definir as permissões ao nível de grupo, de forma que, se liberarmos o acesso de leitura para o grupo professores, por exemplo, todos os usuários que fizerem parte desse grupo terão permissão de leitura nesse arquivo.
  - **Outros**: Simplesmente, todos os usuários que não são o dono e nem pertencem ao seu grupo primário.

Tipos de Permissões de acesso

- Quanto aos tipos de permissões que se aplicam ao dono, grupo e outros usuários, temos 3 permissões básicas:
  - ***r*** (***r****ead*)- Permissão de leitura para arquivos. Caso for um diretório, permite listar seu conteúdo
  - ***w*** (***w****rite*) - Permissão de gravação para arquivos. Caso for um diretório, permite a gravação de arquivos ou outros diretórios dentro dele
  - ***x*** (*e****x****ecute*)- Permite executar um arquivo (caso seja um programa executável). Caso seja um diretório, permite que seja possível aceder através do comando ***cd***
- Estas permissões, **r**, **w** e **x**, não funcionam isoladamente, ou seja, de forma que o usuário tenha ou permissão de leitura ou de gravação ou de execução, elas funcionam em conjunto.

Etapas para acesso a um arquivo/ diretório

- O acesso a um arquivo/diretório é feito verificando primeiro se o usuário que acessará o arquivo é o dono, se for as permissões de dono são aplicadas
- Caso não seja o dono do arquivo/diretório, é verificado se ele pertence ao grupo correspondente do arquivo se for as permissões de grupo são aplicadas
- Caso não pertença ao grupo, são verificadas as permissões de acesso para os outros usuários

Comando chmod

- O comando **chmod** troca as permissões de um arquivo ou directório.
- SINTAXE

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

- Para especificar as novas permissões são usadas as seguintes modos :
  - **O modo literal**, neste modo as permissões podem ser descritas parcialmente, apenas para grupos ou usuários, ou apenas por uma permissão especifica, por exemplo, adicionar permissão de leitura ao grupo.

```
$chmod g+r arquivo.txt
```

  - **E o modo octal**, em que a descrição da permissão é feita de forma completa,

```
$chmod 744 arquivo.txt
```

Comando chmod

Notação Textual

- A notação textual é mais flexível, e pode ser escrita de seguindo a forma, como se pode ver abaixo:

```
$ chmod (ugoa)(+-=)(rwx) arquivos
```

- O primeiro grupo de caracteres refere-se a quem a permissão será alterada
  - **u(u**ser**) -** permissão do usuário proprietário;
  - **g(g**roup**) -** permissão do grupo;
  - **o(o**ther**) -** permissão dos demais usuários;
  - **a(a**ll**) -** permissão de todos os usuários, ou seja, proprietário, grupo e outros usuários.

Comando chmod

Notação Textual (cont.)

- O segundo grupo de caracteres refere-se ao tipo de alteração que será feita.
  - + Adiciona uma permissão;
  - - Remove uma permissão;
  - = Define estado final da permissão para aquele tipo de usuário;
- O terceiro grupo de caracteres refere-se ao tipo de permissão propriamente dita **(r,w,x,-)**
- Exemplos:
  - **chmod g+r** * Permite que todos os usuários que pertençam ao grupo (g) dos arquivos tenham (+) permissões de leitura (r) em todos os arquivos do diretório actual.
  - **chmod uo+x teste.txt** Inclui (+) a permissão de execução do arquivo teste.txt para o dono e outros usuários do arquivo.

Comando chmod

Notação Textual (cont.)

- Exemplos (Cont.)
  - **chmod o-r teste.txt** Retira (-) a permissão de leitura (r) do arquivo teste.txt para os outros usuários (usuários que não são donos e não pertencem ao grupo do arquivo teste.txt).
  - **chmod a+x teste.txt** Inclui (+) a permissão de execução do arquivo teste.txt para o dono, grupo e outros usuários.
  - **chmod u+r,o-w teste.txt** Adiciona (+) a permissão de leitura (r) do arquivo teste.txt para o dono e, retira (-) a permissão de escrita (w) do arquivo teste.txt para os outros usuários.
  - **chmod a=rw teste.txt** Define a permissão de todos os usuários exatamente (=) para leitura e gravação do arquivo teste.txt.

Comando chmod

Notação Octal

- Basicamente deve se pensar no sistema numérico binário, substituindo os tipos de permissões por **0s** ou **1s**. Se determinada permissão e habilitada, atribui-se valor **1(um)**, caso contrario, atribui-se o valor **0(zero)**. Assim, a *string* de permissões ***r-x*** na forma numérica torna-se ***101***. Essa combinação de 1's e 0's é um número binário.
- Depois temos que converter a forma binária para a forma decimal (ou seja, representados por números de 0 a 9).

Comando chmod

Notação Octal

String	Binário	Decimal	Permissões
---	000	0	Nenhuma permissão
--x	001	1	Executar
-w-	010	2	Escrever
-wx	011	3	Escrever e Executar
r--	100	4	Ler
r-x	101	5	Ler e Executar
rw-	110	6	Ler e Escrever
rwx	111	7	Ler, Escrever e Executar

Comando chmod

Notação Octal

- Para exemplificar, vamos utilizar a permissão ***rw-***, cujo valor em binário é ***110***, que por sua vez, em decimal corresponde ao numero *6*. Então, em vez de usar ***rw-*** ou ***110*** para criar a permissão, simplesmente usa-se o numero ***6***.
- Repare que, com o método numérico, usamos somente um digito para representar um conjunto de permissão, em vez de três. Assim, a string de permissões ***r--r--r--*** pode ser representa por ***444***, pois ***r--*** em decimal e igual a quatro.
- Existe outra maneira de pensar:

Permissão	r	w	x	-
Valor	4	2	1	0

Comando chmod

Notação Octal

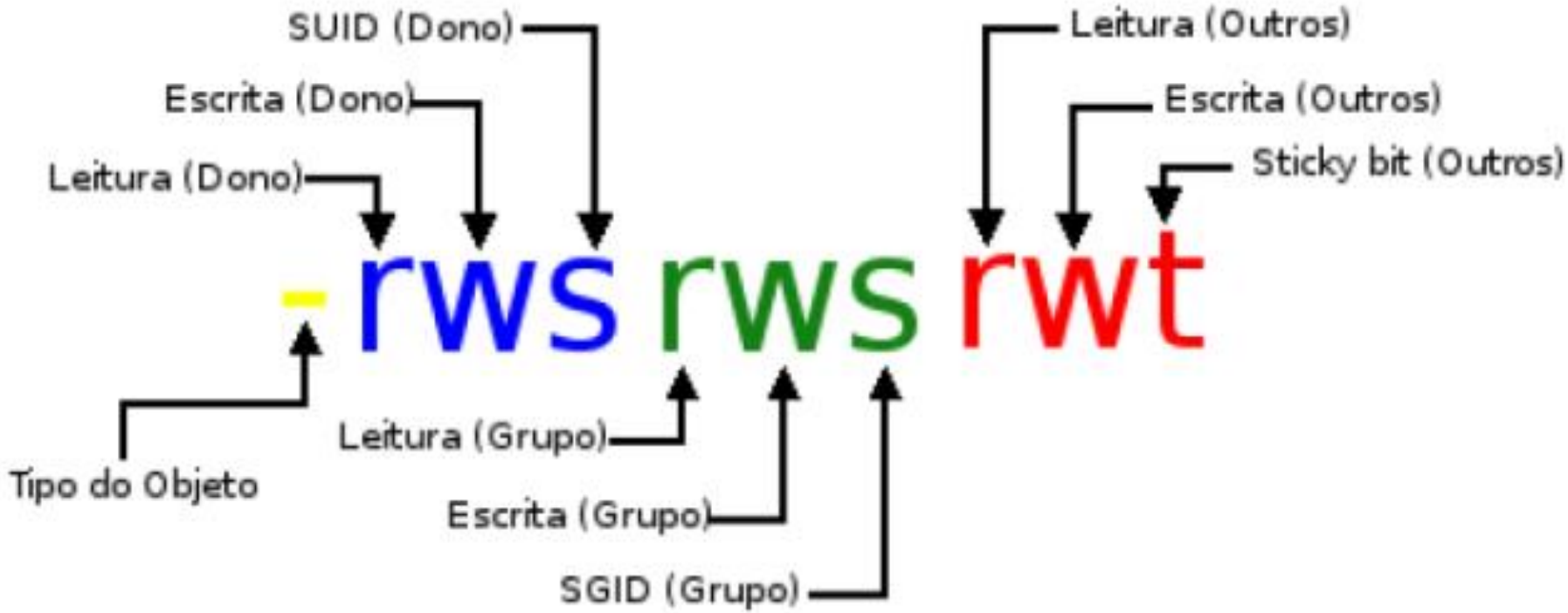
- Exemplos:
  - **chmod 764 teste.txt**

    Neste exemplo temos permissão de leitura, gravação e execução (7) para o dono do arquivo *teste.txt*, temos permissão para leitura e gravação (6) para os membros do grupo do arquivo e permissão de apenas leitura (4) para os outros usuários do sistema.
  - Os números octais são interpretados da direita para a esquerda, portanto o comando **chmod 64 teste.txt.**

    Define permissão de leitura (4) para os outros usuários do sistema e permissão de leitura e gravação (6) para os usuários do mesmo grupo do arquivo *teste.txt*. Neste caso, como as permissões do dono não estão definidas, o sistema assume zero (não tem permissão para leitura, escrita e execução).

Permissões especiais

- Além das permissões simples de leitura, escrita e execução, existem outras três permissões, conhecidas como permissões especiais por só poderem ser aplicadas a tipos específicos de arquivos, que fornecem um alto nível de segurança ao sistema e privacidade aos usuários.
  As permissões especiais são fornecidas pelos chamados bits, a saber: SUID **(Set User ID),** SGID **(Set Group ID)** e STICKY **(Stick BIT)**.



Permissões especiais (cont.)

- Note que estas permissões se referem a execução de um arquivo ou a autorização de acesso a um diretório. Portanto, elas estão relacionadas a permissão **x**. Os seguintes valores são usados para identificar o uso de permissão especial em um arquivo/diretório.

Valor Octal	Valor Binário ugo	Significado
0	000	Nenhuma permissão especial
1	001	Sticky bit
2	010	SGID
3	011	SGID e sticky bit
4	100	SUID
5	101	SUID e sticky bit
6	110	SUID e SGID
7	111	SUID, SGID e sticky bit

Permissões especiais

SUID (Set User ID)

- A propriedade SUID é somente para arquivos executáveis e não tem efeito em directórios.
- No arquivo executável com a propriedade SUID aplicada, o programa será executado com o ID do dono do arquivo, não com o ID do utilizador que executou o programa. Normalmente o utilizador dono do programa executável é também dono do processo sendo executado.
- A permissão de acesso especial SUID aparece no campo Dono

SUID (Set User ID) cont.

- Exemplos:
  - Aplicando a propriedade SUID num ficheiro executável utilizando formato simbólico (s):
    # chmod u+s programa.sh
    # ls -lh programa.sh
    -rwsr-xr-x 1 root roberto 0 2006-09-26 22:57 programa.sh
  - Aplicando a propriedade SUID num ficheiro executável utilizando formato octal para obter as mesmas permissões:
    # chmod 4755 programa.sh
    # ls -lh programa.sh
    -rwsr-xr-x 1 root roberto 0 2006-09-26 22:57 programa.sh

Permissões especiais

SUID (Set User ID) cont.

- Observe que no formato literal, utilizamos u+s para activar (+) a permissão especial de execução (s) para os usuários (u).
- Já no formato utilizamos **4 dígitos** para o comando chmod no formato octal. O primeiro dígito, que até então não tínhamos utilizado, é justamente aquele utilizado para as permissões especiais. **O número 4 corresponde ao bit SUID.**
  - Retirando a propriedade SUID:
    # chmod u-s programa.sh
    # ls -lh programa.sh
    -rwxr-x--- 1 root roberto 0 2006-09-26 22:57 programa.sh
  - Procurando a propriedade SUID num arquivo executável utilizando formato simbólico (s):
    # find /home/roberto -perm u=s
    /home/roberto/programa.sh

Permissões Especiais

SGID (Set Group ID)

- A propriedade SGID funciona com directórios.
- Quando SGID é aplicado em um directório, os novos ficheiros que são criados dentro do directório assumem o mesmo ID de Grupo do directório com a propriedade SGID aplicado.
- A permissão de acesso especial SGID aparece no campo Grupo
- Exemplos:
  - Aplicando a propriedade SGID em um directório */home/equipa* utilizando formato simbólico (s):
    # chmod g+s /home/equipa
    # ls -lah /home/equipa
    drwxr-sr-x 2 roberto equipa 48 2006-09-26 23:21 .

Permissões especiais

SGID (Set Group ID) cont.

- Exemplos:
  - Aplicando a propriedade SGID em um directório */home/equipa* utilizando formato octal:
    # chmod 2755 /home/equipa
    # ls -lah /home/equipa
    drwxr-sr-x 2 roberto equipa 48 2006-09-26 23:21 .
- Note no formato literal, utilizamos g+s para activar (+) a permissão especial de execução (s) para o grupo (g).
- Já No formato octal, veja que o **número 2 corresponde ao bit SGID**, enquanto os outros 3 números correspondem às permissões normais do arquivo.

SGID (Set Group ID) cont.

- Exemplos:
  - Retirando SGID:
    # chmod g-s /home/equipa
    # ls -lah /home/equipa
    drwxr-xr-x 2 roberto equipa 48 2006-09-26 23:21
  - Procurando a propriedade SGID em um directório utilizando formato simbólico (s):
    # find /home -perm /g=s
    /home/equipa

Permissões Especiais

STICKY BIT

- A permissão especial Sticky, pode ser especificada somente no campo outros utilizadores de permissões de acesso.
- Quando activada em diretórios, o usuário pode criar, alterar e remover, apenas os seus próprios arquivos no diretório que possui esta permissão, mesmo que não seja o dono do diretório e nem membro do grupo ao qual o diretório pertence.
- Quando o sticky bit é usado em um arquivo, significa que este arquivo é compartilhado por vários usuários.
- Exemplos:
  - Aplicando a propriedade Sticky em um ficheiro executável utilizando formato simbólico (t):
    # chmod o+t programa.sh
    # ls -lh programa.sh
    -rwxr-x--t 1 root root 2,9M 2006-09-26 23:51 programa.sh

Permissões Especiais

STICKY bit (cont.)

- Exemplos:
  - Para activar o bit STICKY num diretório, utilizando o comando chmod no formato octal com o mesmo efeito.
    # chmod 1755 programa.sh
    # ls -lh programa.sh
    -rwxr-x--t 1 root root 2,9M 2006-09-26 23:51 programa.sh
- Observe que o bit "x" das permissões dos outros usuários foi substituído por um "t", indicando que o bit STICKY está activado.
- No formato octal, **o número 1 corresponde ao bit STICKY**, enquanto os outros 3 dígitos correspondem às permissões simples do diretório.
- No formato literal, utilizamos o+t, para ativar (+) a restrição da exclusão de arquivos (t) para os outros usuários (o).

Comando chgrp

- O comando **chgrp** muda o grupo proprietário do arquivo/diretório.
- SINTAXE:

```
chgrp [opções] grupo arquivo/diretório
```

- Onde:
  - **grupo -** novo grupo que será proprietário 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/sub-diretórios do diretório actual.

Comando chown

- O comando **chown** muda dono de um arquivo/diretório. Opcionalmente pode também ser usado para mudar o grupo.
- Sintaxe:
  chown [*opções*] [dono.grupo] [diretório/arquivo]
- Onde:
  - **dono.grupo -** nome do *dono e do grupo* que será atribuído ao *diretório/arquivo*. O grupo é opcional.

    O *dono.grupo* pode ser especificado usando o nome de grupo ou o código numérico correspondente ao grupo (GID).
  - **diretório/arquivo -** diretório/arquivo que o dono.grupo será modificado.

Comando chown (cont.)

- ***Opções***

  -v, –verbose Mostra os arquivos enquanto são alterados.

  -f, –supress Não mostra mensagens de erro durante a execução do programa.

  -c, –changes Mostra somente arquivos que forem alterados.

  -R, –recursive Altera dono e grupo de arquivos no diretório atual e sub-diretórios.

- É necessário ter permissões de escrita no diretório/arquivo para alterar seu dono/grupo.

Comando chown (cont.)

- Exemplos:
  - **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.
  - **chown -R gleydson.focalinux** * - Muda o dono/grupo dos arquivos do diretório actual e sub-diretórios para gleydson/focalinux (para efeito é necessário ter permissões de escrita no diretórios e sub-diretórios).