

Capítulo 8 - Administração de usuários e permissões - parte 2

8.1. Objetivos

Entender melhor as permissões e caso prático



Um outro modo para setar permissões em um arquivo ou diretório é o octal, que é o mais pedido na LPI:

1 - execução (x)

2 - gravação (w)

4 - leitura (r)

Seguindo a mesma lógica, mas agora usando números. Sabendo o valor das 3 permissões (rwx), precisamos apenas setar as permissões para as 3 (ugo) pessoas. Esses comandos fazem a mesma coisa:



```
#chmod 764 arquivo
```

```
#chmod u=rwx,g=rw,o=r arquivo
```

É importante lembrar que você tem que ser dono de um arquivo/diretório para dar permissão a ele ou ser o root. Quando se cria um arquivo ou mesmo um diretório, ele já vem com uma permissão.



Mas como acontece isso? De onde vem essas permissões?

Existe uma variável no sistema que é responsável por armazenar a permissão padrão é: UMASK. Uma variável tem como função guardar um valor para que esse valor possa ser usado posteriormente por algum programa etc. Além de ser uma variável, UMASK também é um comando:



```
#umask
```

Ela é armazenada nos arquivos profiles (perfis), ou seja, no /etc/profile e no /home/"usuários"/.bashrc que fica no home de cada usuário do sistema. No UMASK vem um valor para gerar a permissão padrão para arquivos e diretórios criados.

E cada usuário pode ter o seu padrão, ou seja, posso alterar o UMASK apenas de um determinado usuário pelo seu arquivo `.bashrc`. Caso a variável não exista em seu `.bashrc`, pode ser criada!

Seu valor padrão é 0022, todo usuário tem seu UMASK:



```
$ cat /home/leo/.bashrc | grep umask  
umask 0022
```

```
# cat /root/.bashrc | grep umask  
umask 0022
```

Se você não encontrar essa linha no seu arquivo `.bashrc`, acrescente-a no final do arquivo. Exemplo:



```
umask 0022
```



O valor dele é 022, mas que permissão esse valor gera? E é exatamente isso que a LPI lhe pede, isto é, calcular o valor de UMASK.

Para calcular o valor de umask: Vejo a permissão que eu quero e tiro do valor da permissão total.

Permissão total é: 777

777

-755

022

Então, temos permissão total ($rw\text{x} = 4+2+1 = 7$)

A regra é simples: Sempre pega-se o valor total de permissão possível, que é 777. Depois pega-se o valor de umask que tem-se, o padrão é 022. Pega-se o valor total e subtrai dele o valor da umask, com isso chega-se no valor da permissão padrão para diretórios.



Uma observação importante: Quando falo permissão total, estou falando de `rwX` (7).

Em diretório estou falando:

r - Posso listar o conteúdo do mesmo

w - Posso criar arquivos dentro do mesmo

x - Posso entrar nele para criar os arquivos ou listar.

Em arquivo estou falando:

r - Posso ler o conteúdo desse arquivo

w - Posso alterar o conteúdo desse arquivo

x - Posso executar esse arquivo.



O sistema por padrão não adota que todo arquivo criado será um shell script (ou seja, um executável). Então a opção x em arquivo não tem que ser setada por padrão, senão terei vários arquivos executáveis que na verdade são apenas arquivos de texto normal. Baseado nisso, o sistema sempre irá tirar a permissão de execução dos arquivos criados.

Então repetindo a conta:

777 - Permissão total

022 - UMASK

755 - Permissão equivalente ao valor de umask 022 para diretório

-111 - Tirando o x que vale 1 das 3 pessoas

644 - Permissão equivalente ao valor de umask 022 para arquivos.

Testando:

Em diretório = 755:



```
#cd /tmp
#mkdir teste
#ls -l
drwxr-xr-x 1 root root teste
```

Em arquivo = 644:



```
#touch arquivo
# ls -l
-rw-r--r-- 1 root root arquivo
```



Não é aconselhável mudar o valor de UMASK sem ser no home de um usuário, pois isso pode afetar o sistema. Mudando o UMASK no /etc/profile, estou falando que qualquer arquivo criado pelo sistema mesmo, terá outra permissão! Então mudem UMASK "APENAS" de Usuários!



Baseado na permissão 777 para um diretório, qual seria o valor de UMASK para essa permissão?

Então vamos a conta:

777 - Permissão total

-777 - Permissão que eu quero

000 - Valor de Umask



Sabendo que o valor de UMASK é 007 qual seria a permissão para um arquivo simples? Vamos a conta:

777 - Permissão total

-007 - Valor de Umask

770 - Permissão equivalente ao valor de umask 007 para diretórios

-111 - Tirando o x (1) pois ele pediu arquivo simples

660 - Permissão para um arquivo simples!

Resposta = 660



Quando usa-se o comando `ls -l` ele mostra o dono do arquivo e o grupo do mesmo. Agora, como alterar essas 2 informações, isto é, trocar o dono do arquivo e o grupo dele?

O root cria um arquivo no /tmp, conseqüentemente ele será o dono desse arquivo, e o grupo também será o seu primário (root), mas se for necessário que as pessoas do grupo selecaobrasileira acesse esse arquivo também, então é preciso tirar o grupo (root) do arquivo e colocar o grupo selecaobrasileira no lugar. Para isso usa-se o comando `chown`:



```
# cd /tmp
# touch arquivo_publico.txt
# ls -l arquivo_publico.txt
-rw-r--r-- 1 root root 0 2004-11-16 14:14 arquivo_publico.txt
# chown root.selecaobrasileira arquivo_publico.txt
ou
# chgrp selecaobrasileira arquivo_publico.txt
```

Ficando assim:



```
# ls -l
-rw-r--r-- 1 root selecaobrasileira 0 2004-11-16 14:14 arquivo_publico.txt
```

Onde a sintaxe do comando sempre será:



```
chown dono.grupo nome(arquivo ou diretório)
```



A vantagem do `chown` em relação ao `chgrp` é que o primeiro altera o proprietário também. Só o ROOT tem esse poder de dizer quem vai ou não ser o dono ou o grupo do arquivo, pois ele é o administrador.

8.2. Prática dirigida: Modelo de como aplicar o conhecimento de permissões no dia-a-dia

Serão mostradas algumas coisas práticas mostrando todos os comandos passo a passo. Esse exemplo prático é sobre o que aprendemos e sobre permissões especiais.

Para fazer os exercícios, é interessante você estar com 4 terminais abertos para não se perder ou ficar confuso, mas isso não é obrigatório, pois é possível fazer isso com um único terminal! Vamos fazer em terminais em modo texto, isto é, 4 shells (TTY1, TTY2, TTY3 e TTY4).

8.2.1. Fase de criação

Para acessar o terminal modo texto 1 (tty1) faça:

```
CTRL + ALT + F1
```

Para acessar o terminal modo texto 2 (tty2) faça:

```
CTRL + ALT + F2
```

Para acessar o terminal modo texto 3 (tty3) faça:

```
CTRL + ALT + F3
```

Para acessar o terminal modo texto 4 (tty4) faça:

```
CTRL + ALT + F4
```

Para voltar ao modo gráfico:

```
CTRL + ALT + F7
```

Imagine que temos vários usuários na máquina e eles irão acessar um diretório público, que terá os documentos de uma determinada área da sua empresa. Então, existirão usuários que poderão acessar esse diretório, e existirão usuários que não poderão. Quando fala-se de diretório compartilhado, o local correto para armazená-lo (não é obrigatório) seria no /mnt.

Pode-se ter uma partição reservada para /mnt, o que seria o mais interessante, pois assim consegue-se deixá-la independente do sistema e ao mesmo tempo limita-se seu tamanho com cotas de disco. No primeiro terminal estaremos como root. Lembre que vai ser usado 4 terminais:

1 - root

2 - Usuário debian

3 - Usuário gnu

4 - Usuário tux

Os usuários debian, gnu e tux serão criados ainda. Então no primeiro terminal como root, crie um diretório público:



```
#cd /mnt  
#mkdir documentos
```

Dando um `ls -ld` nesse diretório veremos que o dono do mesmo é root e seu grupo também é root:



```
# ls -ld documentos/  
drwxr-xr-x 2 root root 4096 Nov 16 14:42 documentos/
```

Agora, crie o grupo que será definido para esse diretório, para que um conjunto de usuários tenha acesso ao mesmo. Criando o grupo diretoria para "setar" no diretório documentos:



```
# groupadd diretoria
```



Lembrando que é apenas um exemplo! Pode ser qualquer nome de grupo!

Verificando se o mesmo foi criado:



```
# cat /etc/group | grep diretoria  
diretoria:x:1001:
```

Com o comando acima, confirma-se que o grupo foi criado e que não tem nenhum usuário que pertence a esse grupo. Ou seja, a linha dele está vazia:



```
diretoria:x:1001: "Não tem nenhum usuário aqui!!!"
```

Apenas criar um diretório e um grupo no sistema. Agora, vá no diretório /mnt/documentos para definir que o grupo desse diretório será diretoria e não mais root:



```
# chown root.diretoria documentos/  
# ls -ld documentos/  
drwxr-xr-x 2 root diretoria 4096 2004-11-16 15:28 documentos/
```



Afinal quando um usuário cria um arquivo ou um diretório na máquina, o mesmo vem com seu grupo particular.

Neste caso, o usuário root que criou o diretório, então, é necessário tirar o grupo root e colocar um público para que seja possível inserir usuários nesse grupo. Assim todos os usuários que pertencerem ao grupo diretoria terão acesso a esse diretório. O dono vai ser o root mesmo, pois como é um diretório público, não tem um único dono. Então, nada melhor que ser o root que administra o sistema.

Resumindo o que fizemos até aqui:

Foi criado o diretório /mnt/documentos



```
#mkdir /mnt/documentos
```

Foi adicionado um grupo no sistema que irá ser público.



```
#groupadd diretoria
```

E setamos que o grupo do diretório documentos será diretoria.



```
#chown root.diretoria documentos/
```



Agora o próximo passo é criar os usuários que irão acessar esse diretório. Que no neste caso será o usuário debian e o gnu. O usuário tux será o que NÃO TERÁ ACESSO! Ou seja, o tux vai ser criado mas não faz parte do grupo diretoria!

Adicionando o Debian:



```
# adduser debian
```

E o usuário GNU agora:



```
# adduser gnu
```

E o usuário TUX agora:



```
# adduser tux
```

As informações ficam ao seu critério. Se quiser pode deixá-las em branco, menos a senha. Verificando se os mesmos existem no passwd:



```
# cat /etc/passwd | grep debian
debian:x:1002:1002:Usuário Exemplo Aula,x,x,x,x:/home/debian:/bin/bash

# cat /etc/passwd | grep gnu
gnu:x:1003:1003:Usuário Exemplo Aula,x,x,x,x:/home/gnu:/bin/bash

# cat /etc/passwd | grep tux
tux:x:1006:1006:Usuário Exemplo Aula,x,x,x,x:/home/tux:/bin/bash
```



Com isso, você já pode abrir os outros 3 terminais, se tornando debian no segundo, gnu no terceiro e tux no quarto. Bom, se eles existem no /etc/passwd então significa que já tenho meus usuários criados!

Mas os mesmos estão com seus respectivos grupos:



```
# groups debian
debian : debian

# groups gnu
gnu : gnu

# groups tux
tux : tux
```



O comando `groups` lista os grupos de um determinado usuário. Cada usuário criado no sistema terá um grupo dele, para que o seu home seja protegido. Então, não vamos mexer no grupo primário deles. O comando `groups` (cai na LPI) lista o grupo de um determinado usuário, sendo que poder ser mais de 1. Um usuário poderá sempre pertencer a quantos grupos precisar.

Então adicione os usuários debian e gnu no grupo diretoria. Lembrando que só o debian e o gnu irão fazer parte desse grupo. O usuário tux está fora.



Como adicionar um usuário em um grupo já criado no sistema?

Adicionando usuário debian ao grupo diretoria:



```
# gpasswd -a debian diretoria
```

Adicionando usuário gnu ao grupo diretoria



```
# gpasswd -a gnu diretoria
```

O comando gpasswd irá cadastrar novos usuários a um determinado grupo. Agora vamos verificar novamente os grupos desses usuários:



```
# groups gnu  
gnu : gnu diretoria
```

Os dois agora fazem parte do grupo diretoria:



```
# cat /etc/group | grep diretoria  
diretoria:x:1001:debian,gnu
```

Resumindo:

Existe um diretório documentos que pertence ao grupo diretoria, e existem 2 usuários que são do grupo diretoria. Tudo isso foi feito no terminal que está o usuário root. A fase de criação acabou.

A permissão padrão para qualquer diretório criado é de 755. Então, os usuários que pertencem ao grupo diretoria, apenas podem ler esse diretório. Faça o teste tentando criar um arquivo dentro do diretório documentos com algum usuário (debian ou gnu).

Lembrando que para criar arquivos vazios é o comando abaixo (tentando criar como usuário debian):



```
$ touch /mnt/documentos/teste.txt  
Não é possível! Permissão negada!
```

Tentando com o usuário gnu:



```
$ touch /mnt/documentos/teste.txt  
touch: cannot touch `/mnt/documentos/teste.txt': Permissão negada
```

Corrija isso!

Então, como eles têm em comum o grupo diretoria, aplica-se as permissões no grupo diretoria. É necessário que os usuários que estão no grupo diretoria possam acessar esse diretório documentos sem problemas dando permissão de gravação (w = 2). Por padrão no GNU/Linux quando um diretório é criado, ele vem com a permissão 755, ou seja, umask 022.

Então, nesse diretório documentos, o acesso total é do dono que no caso é o root. Existe acesso de leitura e poder entrar (execução) para o grupo e o resto dos usuários no sistema. Como vai ser um diretório público, não pode permitir que ninguém mais que não seja o root e as pessoas do grupo diretoria veja ou entre.

Então a permissão para os outros (resto dos usuários do sistema que não é do grupo) já vai ser 0, ou seja, sem permissão nenhuma (o-rx). Já o grupo, que por padrão só tem permissão de ler o que tem dentro do diretório (r leitura, x entrar), é necessário que ele possa gravar lá também. Então o grupo terá 7 (que equivale a rwx) ou ainda g+w. E o dono continuará a ter permissão total. Faça da seguinte maneira:



```
# chmod 770 documentos/  
# ls -ld documentos/  
drwxrwx--- 2 root diretoria 4096 2004-11-16 15:28 documentos/
```

Pode também usar as letras:



```
# chmod u=rwx,g=rwx,o-rwx documentos/  
# ls -ld documentos/  
drwxrwx--- 2 root diretoria 4096 2004-11-16 15:28 documentos/
```

A idéia aqui é ser um diretório reservado para o pessoal da diretoria!

Para quem é da diretoria, o diretório é público, pois todo mundo do grupo poderá ver o de todo mundo do grupo! O restante nem pensar em acessar! Então nosso tux já foi descartado nesse momento.

Tente acesso nesse diretório público com o usuário tux, vá no terminal do tux.



```
$ cd /mnt/documentos/  
bash: cd: /mnt/documentos/: Permissão negada
```

Pronto! Agora tenho os 4 terminais que eu havia falado no início:

1 - root

2 - debian

3 - gnu

4 - tux

Tudo indica que o usuário debian tem permissão de entrar no diretório /mnt/documentos e criar seus arquivos. Teste:



```
$ cd /mnt/documentos/  
$ touch arq_debian.txt  
$ ls  
arq_debian.txt
```

Com o que fizemos, funciona! Agora vamos testar isso com o gnu:



```
$ cd /mnt/documentos/  
$ touch arq_gnu  
$ ls  
arq_debian.txt arq_gnu
```

Com o tux sem chance até de entrar no diretório! Ele ficou totalmente fora! Agora, parece que está tudo certo. Os usuários gravando no diretório. Então, como root analisa-se esse diretório.

8.2.2. Permissão SGID BIT

No terminal 1 como root cheque as coisas.



```
# cd /mnt/documentos/  
# ls -l  
-rw-r--r-- 1 debian debian 2004-11-16 16:47 arq_debian.txt  
-rw-r--r-- 1 gnu gnu 2004-11-16 16:57 arq_gnu
```

Cada usuário está criando os arquivos, virando total dono do mesmo, pois até o grupo é o particular dele. Assim ninguém vai conseguir acessar o de ninguém e o diretório PÚBLICO está mais para PRIVADO. O ideal é que todos os arquivos estejam com o grupo diretoria, pois é o grupo que eles (debian e gnu) tem em comum.

Para corrigir isso, é necessário usar uma permissão especial que se chama sgid bit, que força todos os arquivos de um determinado diretório serem criados com o grupo do diretório e não do usuário que criou. Então, se colocar a sgid bit, significa que o debian e o gnu criarão arquivos com grupo diretoria, pois o diretório documentos tem esse grupo.

Veja como fica essa permissão na prática! E como root, vamos corrija esta falha:



```
# cd /mnt/  
# ls -ld documentos/  
drwxrwx--- 2 root diretoria 4096 2004-11-16 16:57 documentos/  
  
# chmod g+s documentos/  
# ls -ld documentos/  
drwxrws--- 2 root diretoria 4096 2004-11-16 16:57 documentos/
```

Perceba que acrescentou o s em grupo, ou seja, sgid bit, e o sistema trocou o x por s na permissão de grupo, pois é uma permissão especial para grupo, por isso o nome SGID (GID - Identificação de Grupo). Usando o chmod apenas foi acrescentado na permissão que já exista, por isso, o "+". Para especificar permissão sgid bit usando modo octal:

O Sgid bit é o s na permissão de grupo, que equivale ao número 2. Mas para usar número, sete sempre todas as permissões, não é possível acrescentar apenas uma.



```
#chmod 2770 /mnt/documentos
```

Sendo que o 2 na frente diz que é Suid Bit, e 770 o que já tínhamos setado antes. Agora, teste se funciona. Entrando em ação os terminais de usuários. Criando arquivo com o debian:



```
$ cd /mnt/documentos/  
$ touch arq2_debian.txt
```

E criando com o gnu:



```
$ cd /mnt/documentos/  
$ touch arq2_gnu.txt
```

Como root para ver se meu trabalho está bem feito:



```
# cd /mnt/documentos/  
# ls -l arq2*  
-rw-r--r-- 1 debian diretoria 0 2004-11-16 17:16 arq2_debian.txt  
-rw-r--r-- 1 gnu   diretoria 0 2004-11-16 17:16 arq2_gnu.txt
```

Então, pode como root apagar o primeiro arquivo que debian e gnu criaram, pois está errado, valendo agora só o segundo! Veja que o arq estava errado, mas o arq2 dos dois está certo agora! Se você perceber, os arquivos arq2 do debian e do gnu já foram criados da maneira certa, ou seja, respeitando os donos, mas deixando o grupo diretoria.

Quando eu fala-se em diretório público, significa que sempre o meu GRUPO terá que ter permissão de gravação, fugindo do padrão (umask) criado pelo sistema. Se analisar os arquivos, eles estão vindo com a permissão rw-r—r—. Ou seja, mesmo assim o grupo diretoria não tem acesso para gravar no arquivo.

Já que o UMASK é 022, e você pode ver pelos arquivos criados que gera permissão 644. Então, isto significa que um arquivo criado pelo usuário debian só será editado pelo mesmo. Já que o grupo desse arquivo tem apenas permissão 4 (leitura). E não é assim que deve funcionar.

É preciso que qualquer usuário do grupo diretoria, crie arquivos nesse diretório deixando permissão de gravação para o grupo também! Então, é preciso mudar a umask apenas desses usuários. Lembre-se que mexer no valor de UMASK não é uma coisa muito boa quando não aplicamos a um usuário em específico.

Por isso vamos apenas mudar a do debian e do gnu. O valor de umask padrão do sistema e de todos usuários é 022.O que equivale a permissão:

Umask = 0022

Permissão em Diretório = 755

Permissão em Arquivo = 644

Como isto é calculado:

777 - Permissão Total

-022 - Valor de Umask

755 - Permissão para Diretórios

-111 - Tirando o x de execução de arquivos

644 - Permissão para arquivos

Lembrando que o UMASK é o valor da permissão padrão para novos arquivos e diretórios criados por qualquer usuário do sistema. Tenho que definir a UMASK 007 para os usuários.

Umask =007 (Lembre-se que os outros usuários que não pertencem ao grupo não terão nem permissão de leitura, isto é, o diretório é público só para quem está no grupo diretoria)

Permissão em Diretório = 770

Permissão em Arquivo = 660

Como isto é calculado:

777 - Permissão Total

- 007 - Valor de Umask

770 - Permissão para Diretórios

- 111 - Tirando o x de execução de arquivos

660 - Permissão para arquivos

Lembrando que estamos falando de aritmética octal e não decimal. No modo octal o algarismo vai de 0 a 7, enquanto que no decimal vai de 0 a 9.



```
# vi /home/debian/.bashrc  
umask 007 (acrescente ao final do arquivo)
```

Para que a alteração entre em vigor faça:



```
$ source /home/debian/.bashrc
```



O comando source lê novamente os comandos que estão no arquivo. Lembre-se que o .bashrc é lido após o login do usuário.

Depois que a UMASK for alterada deve-se fazer um novo login desses usuários (debian e gnu) ou usar o comando source. Criando um terceiro arquivo como Gnu:



```
$ cd /mnt/documentos/  
$ touch arq3_gnu.txt
```

Como debian agora:



```
$ cd /mnt/documentos/  
$ touch arq3_debian.txt
```

Verificando com o root:



```
# cd /mnt/documentos/  
# ls -ls arq3*  
-rw-rw---- 1 debian diretoria Nov 19 11:16 arq3_debian.txt  
-rw-rw---- 1 gnu   diretoria Nov 19 11:18 arq3_gnu.txt
```

Agora sim, existe a permissão de gravação para o grupo também! Para ficar melhor a compreensão, vai lá como root e apaga os arquivos criados com o arq e arq2, deixando apenas o 3 que está certo. Fique apenas com os arq3.



Dica: UMASK cai na LPI!

Agora antes de complicar mais e para garantir as nossas alterações, crie um 4º arquivo para os dois usuários.



```
Com o gnu:  
$ touch arq4_gnu.txt  
  
Com o debian:  
$ touch arq4_debian.txt
```

Como root:



```
# ls -l
```

```
-rw-rw---- 1 debian diretoria Nov 19 11:16 arq3_debian.txt  
-rw-rw---- 1 gnu   diretoria Nov 19 11:18 arq3_gnu.txt  
-rw-rw---- 1 debian diretoria Nov 19 11:19 arq4_debian.txt  
-rw-rw---- 1 gnu   diretoria Nov 19 11:19 arq4_gnu.tx
```

8.2.3. Permissão STICK BIT

Parece que agora está tudo certo. Então, os usuários estão lá felizes da vida e gravando e compartilhando os arquivos no /mnt/documentos. Lembrando que apenas o debian e o gnu! Então o usuário debian muito amigo:



```
debian@matrix:documentos$ ls  
arq3_debian.txt arq3_gnu.txt arq4_debian.txt arq4_gnu.txt  
debian@matrix:documentos$ rm arq3_gnu.txt  
rm: remove regular empty file `arq3_gnu.txt'? y
```

E o Gnu para se vingar...

```
gnu@matrix:documentos$ ls  
arq3_debian.txt arq4_debian.txt arq4_gnu.txt  
debian@matrix:documentos$ rm arq3_debian.txt  
rm: remove regular empty file `arq3_debian.txt'? y
```

Usuário apagando arquivo de outro usuário. A permissão stick bit em um diretório faz com que apenas o dono e somente ele possa apagar o arquivo que ele e somente ele criou! Essa permissão já vem aplicada no diretório público /tmp. Então para esse problema existe outra permissão especial.

Então vamos no diretório /mnt/ como root arrumar a casa novamente! Antes tínhamos assim:



```
# ls -ld /mnt/documentos/  
drwxrws--- 2 root diretoria 4096 Nov 19 11:34 /mnt/documentos/
```

Agora vamos acrescentar a stick bit que é a letra t:



```
#chmod o+t /mnt/documentos/  
#ls -ld /mnt/documentos/  
drwxrws--T 2 root diretoria 4096 Nov 19 11:34 /mnt/documentos/
```

Sendo que o T tem que estar na permissão do resto dos usuários, que não são donos e nem pertencem ao grupo. Não pode ser na de dono e nem de grupo, pois nenhum outro usuário pode apagar se não for o dono.

Por isso o+t (o = other). Essa permissão tem o valor 1. Então contando que as permissões já estão setadas, é necessário colocá-las tudo novamente quando uso em números:



```
# chmod 3770 /mnt/documentos
```

Sendo que 3 = 2 (Sgidbit) + 1 (Stickbit), e 770 as permissões normais que já foram setadas.



Agora, vá lá nos terminais dos usuários acabar com a farra deles.



O debian tentando apagar o do gnu:

```
debian@matrix:documentos$ rm arq4_gnu.txt
```

```
rm: remove regular empty file `arq4_gnu.txt'? y
```

```
rm: cannot remove `arq4_gnu.txt': Operation not permitted
```

E gnu tentando apagar o do debian:

```
gnu@matrix:documentos$ rm arq4_debian.txt
```

```
rm: remove regular empty file `arq4_debian.txt'? y
```

```
rm: cannot remove `arq4_debian.txt': Operation not permitted
```



Ou seja, só o verdadeiro dono (quem criou) pode apagar! Mesmo que a permissão do arquivo seja total para o grupo!

8.2.4. Permissão SUID

Para demonstrar o uso do SUID, que é outra permissão especial, veja o exemplo do comando shutdown, que é utilizado para desligar e reiniciar o sistema, mas que só pode ser executado pelo usuário root. Mesmo se você der permissão através do "chmod 755 /sbin/shutdown", o usuário comum não vai conseguir realizar a execução deste, somente o root.

Exemplo prático:

Crie um grupo no qual os usuários que poderão reiniciar ou desligar o sistema estarão:



```
# groupadd shutdown
```

O usuário comum leo estará nesse grupo:



```
# gpasswd -a leo shutdown
```

É preciso mudar o grupo do arquivo executável /sbin/shutdown:



```
# chown root.shutdown /sbin/shutdown
```

Aplicando a permissão especial SUID em modo octal, apenas o root terá poder de escrita nesse arquivo:



```
# chmod 4750 /sbin/shutdown
```

Agora, crie um link do arquivo para o /bin. Lembre-se que o usuário comum só executa arquivos que estão em /bin, por isso a necessidade do link. Criando o link:



```
# ln -s /sbin/shutdown /bin/shutdown
```