GNU/Linux para programadores:

Permiso denegado

***Change Mode*, más conocido por su abreviatura `chmod`, es el comando de GNU/Linux que permite modificar los permisos de acceso tanto a archivos como a directorios. En más de una oportunidad te habrás encontrado ejecutando el comando `chmod` seguido de 777 como argumento para solucionar el error "Permiso denegado". Sin embargo, la mayoría de las veces, esa, no siempre es la solución correcta.**

Escrito por: **Eugenia Bahit** (Arquitecta GLAMP & Agile Coach)

Eugenia es **Arquitecta de Software**, **docente** instructora de tecnologías **GLAMP** (GNU/Linux, Apache, MySQL, Python y PHP) y **Agile coach** (UTN) especializada en Scrum y eXtreme Programming. Miembro de la **Free Software Foundation** e integrante del equipo de **Debian Hackers**.

Webs:
Cursos de programación a Distancia: www.cursosdeprogramacionadistancia.com
Web personal: www.eugeniabahit.com

Redes sociales:
Twitter / Identi.ca: **@eugeniabahit**

Cada vez que un alumno ejecuta un `ls -l` y veo que cambió los permisos de todo un directorio Web a 777, en mi cabeza aparece la imagen de *Bruce Dikinson* gritando *"Six, six, six, the number of the beast"*. Claro que 666 no es lo mismo que 777, pero **para el sistema de archivos en GNU/Linux el 777 es** -definitivamente-, **el número de la bestia** ¡créeme!

Nada más frecuente para un programador, que ejecutar su aplicación Web en el navegador y ver el error "Permiso denegado". Pero el mismo, no necesariamente se produce porque un archivo o directorio tenga permisos diferentes a 777. Sin embargo, la primera "solución" que el programador suele encontrar, es cambiar los permisos asignando algo tan "endemoniadamente" peligroso como 777.

Asignar a un archivo o directorio permisos 777, significa asignar permisos de lectura, escritura y ejecución sobre ese archivo o directorio, para cualquier usuario, sea o no el propietario. Pero estos números, no son números al azar, ni mucho menos se necesita andar memorizando números de 3 cifras para entender estos permisos en GNU/Linux. Simplemente, se trata de comprender cómo funciona el sistema de permisos.

Cuando hacemos un listado de archivos y directorios, a la izquierda de cada archivo

(como primera columna) vemos una sucesión de 10 de caracteres:

```
-rw-rw-r--  1 eugenia eugenia  18K mar 29 12:54 Agenda 2013.ods
drwxrwxr-x  2 eugenia eugenia 4,0K ene  5 18:49 certificados
lrwxrwxrwx  1 root root         45 nov  9 18:49 projects -> /home/eugenia/projects
```

A la vez, la tercera columna, muestra el nombre del propietario del archivo (generalmente, usuario que creó el archivo o directorio, o usuario que ha sido asignado como "dueño" por el creador del archivo):

```
-rw-rw-r--  1 eugenia eugenia  18K mar 29 12:54 Agenda 2013.ods
drwxrwxr-x  2 eugenia eugenia 4,0K ene  5 18:49 certificados
lrwxrwxrwx  1 root root         45 nov  9 18:49 projects -> /home/eugenia/projects
```

La cuarta columna indica el grupo al cuál el archivo o directorio pertenece:

```
-rw-rw-r--  1 eugenia eugenia  18K mar 29 12:54 Agenda 2013.ods
drwxrwxr-x  2 eugenia eugenia 4,0K ene  5 18:49 certificados
lrwxrwxrwx  1 root root         45 nov  9 18:49 projects -> /home/eugenia/projects
```

Volviendo a la primera columna (la que muestra la sucesión de 10 caracteres), se subdivide (aunque imaginariamente) en 4 bloques. El primero, de un solo *caracter* y los 3 restantes, de 3 caracteres cada uno:

```
-  rw-  rw-  r--  1 eugenia eugenia  18K mar 29 12:54 Agenda 2013.ods
d  rwx  rwx  r-x  2 eugenia eugenia 4,0K ene  5 18:49 certificados
l  rwx  rwx  rwx  1 root root         45 nov  9 18:49 projects -> /home/eugen...
```

El **primer bloque** de un *caracter*, indica simplemente el **tipo** de elemento del cuál se trata:

```
-          Archivo
d          Directorio
l          Enlace simbólico
```

Los 3 caracteres de cada uno de los **bloques restantes**, indican los **permisos** de acceso:

```
r          (read)      Lectura
w          (write)     Escritura
x          (eXecution) Ejecución
-          sin permiso
```

Esos bloques de permisos, corresponden al propietario, al grupo y a cualquier otro usuario respectivamente:

```
   propietario  grupo  otros
-  rw-          rw-    r--  1 eugenia eugenia  18K mar 29 12:54 Agenda 2013.ods
d  rwx          rwx    r-x  2 eugenia eugenia 4,0K ene  5 18:49 certificados
l  rwx          rwx    rwx  1 root root         45 nov  9 18:49 projects -> /ho...
```

Si tomamos el primer archivo, estamos diciendo que:

```
rw- El propietario tiene permisos de lectura y escritura pero no de ejecución
rw- El grupo tiene permisos de lectura y escritura pero no de ejecución
r-- Otros usuarios solo tienen permisos de lectura
```

En cambio, el tercer archivo (enlace simbólico) tiene permisos de lectura, escritura y ejecución para el propietario, el grupo y para cualquier otro usuario:

```
Propietario: rwx (read, write, execute)
Grupo:       rwx (read, write, execute)
Otros:       rwx (read, write, execute)
```

Las letras que representan cada uno de los permisos, se encuentran asociadas a un número:

```
r   read     lectura    4
w   write    escritura  2
x   execute  ejecución  1
```

Regla para memorizar la asociación de números y letras/permisos: **Cuanto mayor es el número, menor es el permiso.**

Cada bloque de 3 letras, es representado por la suma de los números de cada letra. Es decir que sumando los permisos de cada bloque, se obtiene el "número final" de permiso:

```
r--r--r-- [4]     [4]   [4]          444
-w--w--w- [2]     [2]   [2]          222
rw-rw-rw- [4+2]   [4+2] [4+2]        666
rw-r--r-- [4+2]   [4]   [4]          644
rwxrw-r-- [4+2+1] [4+2] [4]          764
```

Antes de modificar los permisos de un archivo o directorio, se debe pensar previamente en qué necesidades tendrá cada uno de los bloques. Por ejemplo:

```
Propietario: permisos absolutos                  rwx = 4 + 2 + 1 = 7
Grupo:       permisos de lectura y escritura     rw- = 4 + 2 + 0 = 6
Otros:       permisos de lectura                 r-- = 4 + 0 + 0 = 4
```

Un error frecuente, es asignar permisos absolutos (777) para que un usuario en particular, distinto al propietario, pueda tener un acceso completo al archivo:

```
$ ls -l
-rw-rw-r--  1 eugenia developers  18K mar 29 12:54 archivo.foo

$ chmod 777 archivo.foo
$ ls -l
-rwxrwxrwx  1 eugenia developers  18K mar 29 12:54 archivo.foo
```

Pero en el caso anterior, si solo se quiere que, por ejemplo, el usuario "`juan`" tenga un acceso completo, se debe cambiar el propietario del archivo y no los permisos:

```
$ ls -l
-rw-rw-r--  1 eugenia developers  18K mar 29 12:54 archivo.foo

$ chown juan archivo.foo
$ ls -l
-rw-rw-r--  1 juan developers  18K mar 29 12:54 archivo.foo
```

De esta forma, el usuario `juan` por ser propietario del archivo, tendrá acceso para leerlo y escribirlo y el usuario `eugenia`, por pertenecer al grupo `developers`, tendrá el mismo acceso al igual que cualquier otro usuario del grupo `developers`. La alternativa, también podría haber sido incorporar al usuario `juan` al grupo `developers` pero ya nos estaríamos yendo del tema.

Si en cambio, solo se quisiera que los usuarios `juan` y `eugenia` tuviesen acceso al archivo, pero solo `juan` pudiese escribir en él, se coloca como propietario a `juan` y como grupo, uno al que solo pertenezca el usuario `eugenia` y luego, se modifican los permisos para dicho grupo:

```
$ ls -l
-rw-rw-r--  1 eugenia developers  18K mar 29 12:54 archivo.foo

$ chown juan:grupo_eugenia archivo.foo
$ ls -l
-rw-rw-r--  1 juan grupo_eugenia  18K mar 29 12:54 archivo.foo

# chmod 640 archivo.foo
$ ls -l
-rw-r--r--  1 juan grupo_eugenia  18K mar 29 12:54 archivo.foo
```

Pero el cambio de propiedad, **no debe aplicarse a usuarios como** por ejemplo, el usuario de Apache **`www-data`**. Cuando a este usuario se le otorga la propiedad de un

archivo, debe entenderse que ***se está otorgando la propiedad a cualquier persona que acceda a dicho archivo mediante Apache***. Por ejemplo, si se tratara del archivo `foo.html` al cual se accede mediante `http://www.example.org/foo.html`, cualquier persona que accediera a dicha URL, será considerada propietaria de ese archivo, ya que el mismo, será ejecutado por el usuario `www-data`.

Siempre hay que tener en cuenta que además del súper usuario, el propietario de un archivo es quien puede realizar cambios sobre éste, como por ejemplo, cambiar la propiedad del mismo o modificar sus permisos de acceso.

Entonces **¿qué hacer si se necesita que el usuario `www-data` tenga un acceso especial a determinados archivos?**

En principio, al usuario `www-data` se lo debe considerar en el tercer grupo de "cualquier otro usuario". Que el usuario `www-data` sea el usuario de Apache, no significa que solo Apache utilizará dicho usuario. Indirectamente, cualquier persona que realice una petición al servidor, lo estará haciendo en nombre del usuario `www-data`.

Luego, hay que analizar con cautela, qué necesidades reales tendrá este usuario. Por defecto, la única necesidad real de los archivos accedidos por el usuario `www-data` es la de lectura. Para los directorios, se suma la de ejecución a fin de que se pueda ingresar en ellos. Por lo tanto, **un directorio web debería tener los siguientes permisos**:

- Propietario: lectura, escritura y ejecución: 4 + 2 + 1 = 7
- Grupo: no son necesarios permisos: 0 (aunque una buena práctica, es asignar los mismos permisos al grupo que al resto de los usuarios)
- Resto de los usuarios (incluye a www-data): lectura y ejecución: 4 + 1 = 5

```
chmod 705 directorio_web # Opción 1: Grupo sin permisos
chmod 755 directorio_web # Opción 2: Grupo mismos permisos que el resto de usuarios
```

Luego, los archivos tendrán los mismos permisos que el directorio, menos el de ejecución para el resto de usuarios:

```
chmod 704 archivo_web # Opción 1: Grupo sin permisos
chmod 744 archivo_web # Opción 2: Grupo mismos permisos que el resto de usuarios
```

Asignando 705 a los directorios y 704 a los archivos, "Permiso Denegado" sería un error que jamás deberías volver a ver.