Redirección de E/S

Cuando se ejecuta un comando en Linux, se abren tres corrientes de datos estándar: standard input (stdin), standard output (stdout) y standard error (stderr).
Todo comando toma los datos de entrada de stdin y envía su salida normal a stdout y los errores a stderr.
Por defecto, stdin es el teclado mientras que stdout y stderr es la pantalla de la terminal.
Sin embargo, un usuario puede redirigir cualquiera de estas corrientes de datos a un archivo, dispositivo o a algún otro comando. Este proceso se denomina redirección de E/S.

Redirección de stdout

La redirección de stdout se controla con el signo ">". Por ejemplo, si tenemos un archivo file1 y le aplicamos el comando cat, obtenemos lo siguiente:

```
$ cat file1
Primera línea.
Segunda línea.
Tercera línea.
$
```

Como se ve, en este caso stdout apunta a la terminal. Pero si quisiéramos que la salida de file1 se guarde en otro archivo file2, debemos ejecutar el siguiente comando:

```
$ cat file1 > file2
```

El comando precedente no muestra nada en pantalla, porque su salida fue redirigida al archivo file2.
Si vemos el contenido de file2 veremos que contiene el mismo texto que file1.
Utilizando la redirección también podemos unir dos o más archivos en uno. Supongamos que tenemos dos archivos (file1 y file2) que queremos unir en otro archivo (file2). Podemos utilizar el comando cat para mostrar el contenido de los dos archivos y redirigir su salida a otro archivo:

```
$ cat file1 file2 > file3
```

En el caso de la redirección de la salida con el operador ">" el archivo al que se dirige la salida es sobrescrito. Si queremos que la salida sea agregada al final del archivo, podemos usar el operador de doble redirección ">>". Por ejemplo, para concatenar el contenido del archivo file1 con el de file2, se debe ejecutar el siguiente comando:

```
$ cat file1 >> file2
```

Redirección de stdin

Con la redirección de stdin se reemplaza la entrada por defecto del teclado al archivo o medio que se indique. Para esto se utiliza el operador "<". Por ejemplo, consideremos un archivo de texto cuyo contenido no esté ordenado alfabéticamente y que se llame "desordenado".

```
$ cat desordenado
Línea 1
Línea 3
Línea 2
$
```

Podemos utilizar el comando sort redirigiendo el archivo "desordenado" a su stdin de la siguiente manera:

```
$ sort < desordenado
```

```
Línea 1
Línea 2
Línea 3
$
```

Redirección de stderr

stderr también puede redirigirse de un modo similar a stdin y stdout. Para esto se utiliza el operador “2>”. Por ejemplo, si queremos que los errores generados por el comando ls se envíen al archivo error.txt, debemos ejecutar lo siguiente:

```
$ ls xyz 2> error.txt
$
$ cat error.txt
ls: xyz: No such file or directory
$
```

Utilización de pipes

Otra funcionalidad útil de Linux es el pipe (|), con el que se puede enviar la salida de un comando a la entrada de otro. Esto es normalmente útil para procesar y dar formato a los datos producidos por un comando para hacerlo más entendible.
Para esto se utiliza el operador “|” (o pipe). Cuando dos comandos están conectados por un pipe, el primero envía su salida al pipe en vez de enviarlo a la terminal, y el segundo comando lee su entrada del pipe en lugar del teclado. Los pipes se utilizan normalmente para filtrar, modificar o manipular los datos de salida de un comando. Se pueden utilizar múltiples niveles de pipes y se pueden utilizar en combinación con los operadores de redirección.
Como ejemplo podemos tomar el caso en que queremos listar los archivos de un directorio y éstos son tan numerosos que requieren más de una pantalla para poder ser vistos. En este caso podemos utilizar un pipe para redirigir la salida del comando ls al comando more, de la siguiente manera:

```
$ ls /etc | more
```

Esto hará que la salida del comando ls (el listado de archivos) sea paginada por el comando more de modo tal que se pueda ir pasando página a página la lista de archivos.

El editor vi

El editor vi es el más ampliamente utilizado en los sistemas operativos tipo-UNIX. Inicialmente, los usuarios lo encuentran difícil para usar, pero luego, al acostumbrarse a su uso, comienza a gustarles por su simplicidad y poder. Este editor es capaz de manejar múltiples archivos simultáneamente, cortar, copiar y pegar texto, buscar y reemplazar, y hasta chequear la ortografía.

Existen tres modos que se pueden utilizar en vi: el modo de comandos, el modo de última línea y el modo de inserción. El modo de comandos es utilizado para recibir comandos, los cuales no se muestran en pantalla y se ejecutan cuando se presiona una tecla de comandos. El modo de última línea recibe comandos que se muestran en la última línea del editor y comienzan generalmente con dos puntos (:). Y el modo de inserción permite la entrada de texto.
Al arrancar vi, éste se encuentra en modo de comandos. Esto significa que cualquier tecla presionada es considerada un comando por el editor. Para pasar a modo de inserción se debe presionar i en modo comandos. Luego de esto, cualquier tecla que se presione se interpreta como texto y se muestra en pantalla. Para pasar nuevamente a modo de comandos, se debe presionar la tecla Esc. Para crear o editar el archivo file1 con vi se debe ejecutar lo siguiente:

$ vi file1

El caracter tilde (~) significa que el archivo no tiene texto en esas líneas. Para pasar a modo de inserción, se debe presionar la tecla i y a partir de ese momento se puede comenzar a insertar texto al archivo. Una línea de texto puede ser de hasta 256 caracteres de largo. Presionando la tecla Enter se pasa a la siguiente línea, y cuando se termina de ingresar texto, se presiona la tecla Esc para retornar a modo de comandos. Para terminar y salir, se debe presionar la tecla (:), con lo que se pasa a modo de última línea, en la que pueden ingresar distintos comandos como por ejemplo el comando :wq con el que se graba el archivo y se sale del editor.

En la siguiente tabla se muestran los comandos más comúnmente utilizados en vi:

Comando	**Descripción**
Comandos de Movimiento del Cursor	
l	Mover cursor hacia la derecha
h	Mover cursor hacia la izquierda
j	Mover cursor hacia abajo
k	Mover cursor hacia arriba
^^	Mover cursor al comienzo de la línea
$	Mover cursor al final de la línea
G	Mover cursor al final del archivo
nG	Mover cursor a la línea n
Comandos de inserción	
i	Insertar a partir del cursor
I	Insertar al comienzo de la línea actual
a	Insertar en la posición siguiente a la del cursor
A	Insertar al final de la línea actual
o	Agregar una línea abajo de la actual para insertar texto
O	Agregar una línea arriba de la actual para insertar texto
Comandos de Borrado	
x	Borrar caracter en la posición del cursor
nx	Borrar n caracteres a partir de la ubicación del cursor
X	Borrar caracter anterior al cursor
nX	Borrar n caracteres anteriores al cursor
dd	Borrar la línea actual
ndd	Borrar n líneas desde la actual
dG	Borrar líneas hasta el final del archivo
Comandos de reemplazo	
r	Reemplaza el caracter actual
s	Reemplaza el caracter actual y pasa a modo inserción
R	Reemplaza múltiples caracteres hasta que se presione
Esc	
C	Reemplaza hasta el final de la línea
cc	Cambiar la línea entera
Copiar y pegar	
yy	Copiar la línea actual
nyy	Copiar n líneas a partir de la actual
P	Pegar líneas copiadas en la línea siguiente de la actual
p	Pegar líneas copiadas en la línea anterior de la actual
Comandos de modo última línea	
:w	Guardar archivo

:q	Cerrar archivo
:x	Guardar y salir
:wq o :x	Guardar y salir
:q!	Salir sin cambios
/texto1	Buscar texto1
:1,$s/texto1/texto2/g	Buscar texto1 y reemplazar por texto2

EXPRESIONES REGULARES

Las expresiones regulares tienen un potencial increíble, pero a diferencia de los comodines necesitan una mayor esfuerzo por tu parte. Por una lado necesitan de una curva de aprendizaje más o menos pronunciada, y por otro lado, su aplicación no es tan intuitiva como en el caso de los comodines.

A continuación encontrarás algunos de los elementos con los que construir expresiones regulares. Sin embargo, al contrario que con los comodines, dejaré los ejemplos para el final.

.. Representa un solo carácter, es el equivalente a la ? de los comodines.
\. Se utiliza para escapar caracteres, al igual que en el caso de los comodines.
+. El elemento precedente puede aparecer 1 o más veces.
*. Es similar al anterior, pero en este caso, el elemento puede aparecer cero o más veces.
^ ó \A. Dependiendo de su ubicación tiene un significado u otro. Así si lo encuentras al principio se corresponde con que lo que buscamos debe comenzar igual. Si lo encontramos entre corchetes es una negación, como veremos más adelante.
$ ó \Z. Es el opuesto al anterior, en tanto en cuanto, cuando lo encuentras al final indica que lo que buscamos tiene que acabar igual.
[]. Especifica un rango, ya sea separado por comas como [a,b,c] o bien [a-c].
|. Representa un o lógico.
[^]. Como he indicado anteriormente, esta expresión se utiliza para negar rangos.
{}. Indica el número de repeticiones del caracter precedente. Así a{3} es equivalente a aaa. Pero además a{3,} representa 3 ó más a. En el caso de a{1,5} se refiere a entre 1 y 5.
Además de estas que son muy parecidas a los comodines, en el caso de las expresiones regulares tenemos más, como son las siguientes,

\s. Se corresponde con un espacio en blanco.
\S. El contrario del anterior, es decir, cualquier carácter que no sea un espacio en blanco.
\d. Equivale a un dígito.
\D. Cualquier carácter que no sea un dígito.
\w. Se corresponde a cualquier carácter que se pueda utilizar en una palabra. Es equivalente a [a-zA-Z0-9_].
\W. El opuesto al anterior, es decir, es equivalente a [^a-zA-Z0-9_].
Por supuesto, también tiene en cuenta los caracteres escapados, que comenté anteriormente, y entre los que cabe citar,

\n. Se corresponde con una línea nueva.
\r. Para retorno de carro.
\t. Representa una tabulación.
\0. Se utiliza para un carácter nulo
Por último queda el rey que es () cuyo objetivo es el de capturar todo lo que está en su interior.

ALGUNAS EXPRESIONES REGULARES ÚTILES

Normalmente van delimitados por //. Es decir, la expresión regular viene a ser /expresión-regular/.

Caracteres alfabéticos: /^[a-zA-Z]*$/. Estos patrones son los más sencillos de utilizar.
Caracteres en minúsculas: /^([a-z]*$ Igual que en el caso anterior, pero esta vez en minúsculas.
Números: /^[0-9]*$/. La tercera opción pero solo para dígitos. Sin embargo, a partir de ahora, la cosa se va a ir complicando.
Nombre de usuario: /^[a-z0-9_-]{3,16}$/. En este caso, el nombre de usuario debe tener una longitud mínima de tres caracteres y máxima de 16. Solo puede estar compuesto por letras minúsculas, números, además de _ y -. Podrías incluir también las mayúsculas de la siguiente forma, /^[a-zA-Z0-9_-]{3,16}$/.
Password: Esto es muy similar al anterior, pero, por un lado podemos considerar cambiar la longitud mínima y máxima de la contraseña, y por otro lado incluir algunos otros caracteres. Así podría ser algo como [a-zA-Z0-9_-$!¡@?¿=;:]{8,20}$/. Hemos aumentado la longitud mínima a 8 y la máxima a 20 y además hemos incorporado algunos caracteres más o menos extraños como pueden ser$!¡@?¿=;:.
Email: /^([a-z0-9_\.-]+)@([a-z0-9_\.-])\.([a-z\.]{2,6})$/. Se divide en tres grupos diferenciados, tipo antes@despues.fin. Los dos primeros admiten letras, números, además de _, - y .. Mientras que el último solo admite letras y el ., pero además con una longitud mínima de 2 y máxima de 6.
Web: Otro patrón realmente interesante, /^(https?:\/\/)?([\da-z\.-]+)\.([a-z\.]{2,6})([\/\w \.-]*)*\/?$/
Código postal: [0-9]{5}(\-?[0-9]{4})?$
Direcciones IP: /^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$. Con esto nos aseguramos que cada uno de los grupos es número comprendido entre 0 y 255.
Etiquetas html: Esta es muy interesante y seguro que la utilizarás en más de una ocasión. Se trata de un patrón para etiquetas html. /^<([a-z]+)([^<]+)*(?:>(.*)<\/\1>|\s+\/>)$/
Tarjetas de crédito: /^(?:4[0-9]{12}(?:[0-9]{3})?|5[1-5][0-9]{14}|6011[0-9]{12}|622((12[6-9]|1[3-9][0-9])|([2-8][0-9][0-9])|(9(([0-1][0-9])|(2[0-5])))[0-9]{10}|64[4-9][0-9]{13}|65[0-9]{14}|3(?:0[0-5]|[68][0-9])[0-9]{11}|3[47][0-9]{13})*$/.
Fechas: '#^((19|20)?[0-9]{2}[- /.](0?[1-9]|1[012])[- /.](0?[1-9]|[12][0-9]|3[01]))*$#
Imágenes: /[^\s]+(?=\.(jpg|gif|png))\.\2)

SERVICIOS ONLINE DE EXPRESIONES REGULARES

Una herramienta que no puede faltar cuando se trabaja con expresiones regulares es algún servicio online que permita probar las expresiones regulares. Durante confección de una expresión regular, es más que fácil de equivocarse en algún carácter. En este sentido, es muy recomendable recurrir a estos servicios, que ayudan notablemente a conseguir una expresión regular perfecta.
A continuación puedes encontrar algunos de ellos:

- regex101.com
- regexr.com
- RegEx Testing
- freeformatter.com

Más info sobre expresiones regulares en RegexOne.

Comodines y Símbolos . ..

Los comodines se suelen utilizar para realizar operaciones sobre varios archivos. Seguro que en más de una ocasión, has visto has utilizado ls *. Este sencillo comando nos permite listar todos los archivos que se encuentran en el directorio, en el que hemos ejecutado esta instrucción. Los comodines se pueden utilizar con prácticamente cualquier otro comando, mv, cp, rm y muchos otros.

A continuación puedes ver un listado de los comodines mas utilizados,

?: Representa un único carácter. Así por si ejecutas ls /dev/sda? te listará todos los archivos que sean igual a /dev/sda mas un único caracter, que puede ser cualquier letra o número. Así se lista las unidades, de la /dev/sda1 a la /dev/sda7.

: Este comodín representa desde nada, hasta cualquier cantidad de caracteres y dígitos. Así, en el ejemplo indicado anteriormente, si ejecutados ls /dev/sd se listará /dev/sda, /dev/sda1 a /dev/sda7 y si hay otra unidad, /dev/sdb, /dev/sdb1,…

[]: En este caso, este comodín representa un rango, ya sea de caracteres o de números. Siguiendo con el ejemplo anterior, podemos listar ls /dev/sda[1-5].

{}: Esto en realidad es más un conjunto de comodines separados por comas. Igual que en casos anteriores, podrías ejecutar la siguiente orden ls {sda*,sdb*}. No dejar espacios alrededor de la coma, por que dará un error.

[!]: El funcionamiento de este comodín es similar a [], salvo que representa justo lo contrario. Es decir se trata de listar todo aquello que que no esté en ese rango. Por ejemplo, ls /dev/sda[!1-5] te mostrará /dev/sda6 y /dev/sda7.

\: Esta es la secuencia de escape. Con este carácter puedes mostrar otros caracteres. Así, cuando se quiere crear el directorio esta casa y ejecutas mkdir esta casa, te creará dos directorios, esta y casa. Para hacer lo que quieres, tienes diferentes alternativas, o bien mkdir "esta casa" o bien mkdir esta\ casa. En esta segunda orden utilizamos \ para escapar el carácter espacio.

Tipos de archivo

– : Archivo
d : Directorio
b : Archivo de bloques especiales (Archivos especiales de dispositivo)
c : Archivo de caracteres especiales (Dispositivo tty, impresora…)
l : Archivo de vinculo o enlace (soft/symbolic link)
p : Archivo especial de cauce (pipe o tubería)

Permisos de usuarios y grupos

El control de los usuarios y grupos es un elemento clave en la administración de sistemas Linux. Los usuarios pueden ser gente real, es decir, cuentas ligadas a un usuario físico en particular o cuentas que existen para ser usadas por aplicaciones específicas.

Los grupos son siempre expresiones lógicas de organización, reuniendo usuarios para un propósito común. Los usuarios dentro de un mismo grupo pueden leer, escribir o ejecutar archivos que pertenecen al grupo.

Cada usuario y grupo tiene un número de identificación único llamado userid (UID) y un groupid (GID) respectivamente. Cuando se crea un archivo se le asigna a un usuario y a un grupo. De la misma manera se asignan los permisos de lectura, escritura y ejecución para el propietario del archivo, para el grupo y para cualquier otro usuario en un host. El usuario y el grupo de un archivo particular, así como los permisos en ese archivo, pueden ser cambiados por un root o en la mayoría de los casos, por el creador del archivo.

Una de las tareas más importantes de cualquier administrador del sistema, es la de administrar adecuadamente usuarios y grupos, así como asignar y revocar permisos. En esta unidad veremos cómo se manejan los usuarios y los grupos bajo Linux.

Manejo de usuarios

Archivos /etc/passwd y /etc/shadow

El archivo /etc/passwd es un archivo de texto plano en el que se guarda la información relativa a los usuarios del sistema, tal como el home directory, la password, el shell, etc. Por cada usuario del sistema existe una línea en este archivo que tiene el siguiente formato:
nombre_cuenta:password:UID:GID:nombre_completo:home-dir:shell

Nombre de la cuenta: es el nombre que va a tener la cuenta de usuario en el sistema. No debe contener letras mayúsculas.
Password: contraseña encriptada del usuario. Este campo puede también contener un asterisco, lo cual significa que el usuario está bloqueado; o la letra "x", que indica que la password está almacenada en el archivo /etc/shadow.
UID: una identificación numérica única para el usuario.
GID: una identificación numérica única para el grupo primario del usuario.
Nombre completo: en este campo se coloca el nombre completo del usuario real o algún comentario que se quiera agregar.
Home directory: indica el home directory del usuario en cuestión.
Shell: aquí se indica el comando que se va a ejecutar en el momento que el usuario inicie una sesión. Usualmente es el shell.

El archivo /etc/passwd tiene permiso de lectura para todos los usuarios del sistema, lo cual lo hace un tanto inseguro puesto que en él se almacenan las contraseñas de los usuarios. Es por esto que los sistemas Unix modernos(como Linux) utilizan el esquema de claves shadow, las cuales se almacenan en un archivo que sólo es legible por root y que además brinda opciones adicionales como puede ser la expiración de claves.

El archivo /etc/shadow posee el siguiente formato:
nombre_cuenta:password:último_cambio:mínimo:máximo:advertencia:inactivo:expiración:flag

Nombre de la cuenta: nombre de la cuenta de usuario.
Password: contraseña del usuario encriptada de 13 caracteres. Si en este campo hay un signo "!" o "*" significa que el usuario está bloqueado.
Fecha de último cambio: la fecha del último cambio de password se da en el número de días desde el 1 de Enero de 1970.
Mínimo y máximo: la contraseña no debería cambiarse nuevamente hasta que haya transcurrido un mínimo de días, y debe cambiarse después de que pase un máximo de días. Si el número mínimo de días requeridos es mayor que el numero máximo de días permitidos, esta password no puede ser cambiada por el usuario.
Advertencia: número de días que debe avisarse al usuario antes de que su password expire.
Inactividad: número de días después de que expira la password para que se considere la cuenta como inactiva.
Expiración: días desde el 1 de Enero de 1970 en que la cuenta expiró.
Flag: campo reservado.

Manejo de grupos

Archivos /etc/group y /etc/gshadow

La información relativa a los grupos de usuarios se almacena en el archivo /etc/group. Este archivo es similar a /etc/passwd y tiene el siguiente formato:
nombre_grupo:password:GID:usuarios

Nombre de grupo: es el nombre que va a tener este grupo de usuarios.
Password: guarda la contraseña del grupo de usuarios.
GID: identificador numérico único para el grupo en cuestión.
Usuarios: lista de usuarios separados por coma que pertenecen al grupo. Al igual que con el archivo /etc/passwd, existe un archivo /etc/gshadow para el archivo de grupos.

Permisos de archivo

Como en todo sistema operativo Unix o tipo-Unix, Linux asocia un archivo o directorio con un usuario y un grupo. Además, cada archivo puede tener permiso de lectura, escritura y ejecución.

Permiso de lectura significa que el archivo puede ser leído pero no modificado o borrado. Permiso de escritura significa que el archivo puede ser creado, modificado o borrado. Finalmente, permiso de ejecución significa en los archivos que pueden ejecutarse y en los directorios significa que pueden ser abiertos.

Hay tres conjuntos de permisos para cada archivo o directorio: usuario, grupo y el resto del mundo. Para cada conjunto, hay permisos separados de lectura, escritura y ejecución. Los permisos del dueño son para el dueño del archivo o directorio. Los permisos de grupo son para todos los usuarios que pertenezcan a él. Y los permisos globales son para todos. Consideremos un ejemplo:

-rw-rw-r-- 1 peter users 1064 Sep 7 08:34 archivo.txt

drwxr-xr-x 1 root users 1064 Sep 7 08:35 directorio

En esta salida del comando ls –l archivo.txt se puede ver la siguiente información:

Tipo de Información	Salida
Permisos de acceso de archivo	-rw-rw-r--
Numero de links	1
Usuario(dueño del archivo)	Peter
Grupo	Users
Tamaño de archivo(en bytes)	1064
Fecha de última modificación	Sep 7
Hora de última modificación	08:34
Nombre de archivo	archivo.txt

Los permisos se listan en la primera columna. El primer carácter indica si se trata de un archivo o de un directorio. Los próximos tres caracteres indican los permisos para el propietario del archivo, los siguientes tres los permisos del grupo y los últimos tres los permisos de todos los demás. Los permisos de lectura, escritura y ejecución se indican con las letras r, w y x, respectivamente.

El comando estándar en Unix/Linux para fijar permisos en archivos o directorios es chmod, y su sintaxis se muestra a continuación:
chmod [-R] modo archivo(s)_o_directorio(s)

La opción –R significa que el comando se debe aplicar en forma recursiva dentro de un directorio.
Los permisos tienen dos modos de uso, el numérico y el simbólico.

Modo numérico
0 = --- = sin acceso
1 = --x = ejecución
2 = -w- = escritura
3 = -wx = escritura y ejecución
4 = r-- = lectura
5 = r-x = lectura y ejecución
6 = rw- = lectura y escritura
7 = rwx = lectura, escritura y ejecución
Nos permite mediante cuatro octales diferentes permisos. Si se omite uno de ellos se entiende que es cero.
En el caso de directorios la x no representa ejecución, sino que se refiere a acceso al directorio en cuestión.

Modo simbólico
El modo simbólico es una combinación de letras y símbolos, que nos permitirán establecer los permisos para un archivo.
u : es para el usuario
g : para el grupo
o : es para los que no pertenecen al grupo
a : para todos
+ : añade un permiso
- :quita un permiso
= : asigna un permiso
r : lectura
w: escritura
x : ejecución

Para fijar los permisos de un archivo o directorio simplemente se deben sumar estos números. Por ejemplo, 7 significa lectura, escritura y ejecución. Al pasar el modo a chmod se indican los permisos para el usuario, el grupo y el resto de usuarios. Por ejemplo, para hacer que el archivo anteriormente listado tenga todos los permisos, se debería ejecutar el siguiente comando:

chmod 777 archivo.txt

ls –l archivo.txt

-rwxrwxrwx 1 peter users 1064 Sep 7 08:34 archivo.txt

Para poder cambiar los permisos de un archivo o directorio uno debe ser el propietario o root. Otro tema a tener en cuenta es la propiedad de los archivos o directorios. Sólo el usuario root puede cambiar la propiedad de un archivo utilizando el comando chown [-R] nombre_usuario archivo. Por ejemplo, para cambiar la propiedad de archivo.txt a root, grupo root, se debe ejecutar el siguiente comando:

chown root.root archivo.txt

ls –l archivo.txt

-rwxrwxrwx 1 root root 1064 Sep 7 08:34 archivo.txt

Permisos especiales

Existen ciertos permisos especiales que pueden definirse para un determinado archivo o directorio. Estos son el sticky bit(o t-bit), el SUID y el SGID.

Sticky bit

El sticky bit es un permiso especial que puede aplicarse a directorios para hacer que cualquier usuario pueda crear un archivo en el directorio, pero que un usuario no pueda borrar archivos de otro usuario. Un ejemplo donde se aplica esto es en el directorio /tmp.

El número que representa el sticky bit es el 1, y se debe anteponer a los permisos normales que se utilizan en el modo del comando chmod. Por ejemplo, para fijar el sticky bit y todos los permisos a un directorio se debe ejecutar el comando:

chmod 1777 directorio

ls –l directorio

drwxrwxrwt 1 root users 1064 Sep 7 08:35 directorio

SUID, SGID

En ocasiones, usuarios que carecen de privilegios tienen que realizar tareas que requieren de privilegios. Un ejemplo es el programa passwd que le permite a un usuario cambiar su contraseña. Hacerlo requiere cambiar el campo de contraseña en el archivo /etc/passwd. Sin embargo, no es conveniente otorgar al usuario acceso directo a este archivo, puesto que el
usuario podría cambiar la contraseña de todos los demás.

Para evitar estos problemas, Linux permite que los programas tengan privilegios. Los procesos que ejecutan estos programas pueden adquirir un UID o GID distinto al propio cuando se están ejecutando. Un programa que cambia su UID se llama programa SUID (set-UID: se establece el UID) y un programa que cambia su GID se llama programa SGID (set-GID: se establece el GID). Un programa puede ser SUID y SGID al mismo tiempo.

Cuando se ejecuta un programa SUID, su UID efectivo se convierte en el dueño del archivo, en lugar del UID del usuario que lo está ejecutando. Si un programa es SUID o SGID la salida de la instrucción ls –l tendrá la letra “s” en lugar de la letra “x” en donde corresponda. En el caso de que el archivo o directorio no tenga establecido el permiso de ejecución, en vez de s o t minúscula se utiliza la “S” o “T”.

Para representar estos permisos con números al utilizar el comando chmod, se utiliza un 2 para el permiso SGID, y un 4 para el permiso SUID.
Ejemplos:

SUID:
chmod 4755 mi_script
-rwsr-xr-x 1 root users 1064 Sep 7 08:35 mi_script

SGID:
chmod 2755 mi_script
-rwxr-sr-x 1 root users 1064 Sep 7 08:35 mi_script