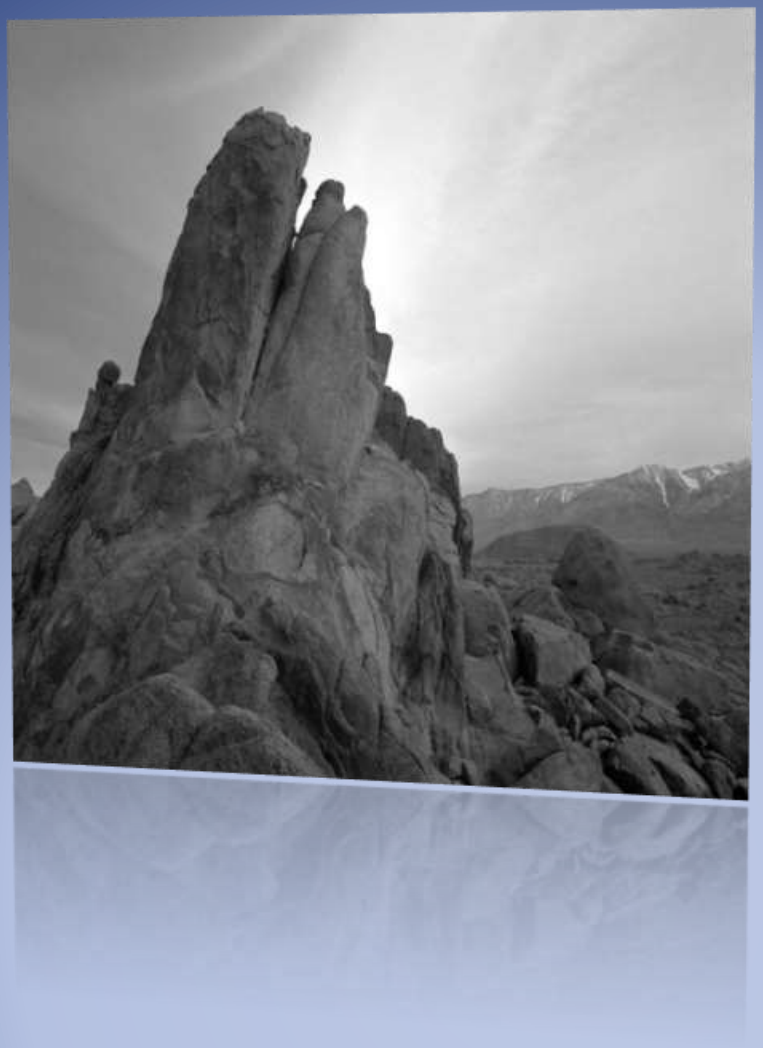


# Gestión de ACLs en redhat

*Redhat para todos*

Breve manual de configuración de ACLs en redhat

## INTRODUCCION - ¿Qué ACLs?

Una de las características que se echan en falta en los sistemas Linux actuales, especialmente en entornos corporativos, es la posibilidad de especificar los permisos de acceso a los ficheros con mayor granularidad. Los sistemas Linux siguen el modelo de permisos Unix tradicional segementando el tipo de acceso en tres categorías:

- El propietario del fichero (User)
- El grupo al que pertenece el fichero (Group)
- El resto de usuarios del sistema que no están en ninguna de las dos categorías anteriores (Other).

También conocido como modelo UGO (User, Group, Other).

Sin embargo, estas tres categorías se revelan insuficientes en una gran cantidad de situaciones, donde desearíamos poder especificar permisos diferenciados para varios usuarios o grupos determinados.

Aquí es donde entran en juego los permisos basados en Listas de Control de Acceso, más conocidos como ACLs. En este sistema de permisos los ficheros no tienen un juego fijo de permisos (como en el modelo tradicional, que tiene tres permisos y sólo tres), sino que los permisos del fichero son en realidad una lista de Entradas de Control de Acceso (o ACEs). Cada una de estas ACEs contiene un par (usuario/grupo, permiso) que indica un tipo de acceso determinado para un usuario o grupo, y el conjunto de todas ellas forman la ACL que marca el tipo de acceso permitido en un fichero.

Este sistema es el utilizado entre otros, por los sistemas de ficheros NTFS (de Windows NT y sucesores), el sistema UFS de Solaris y el sistema HFS de HP-UX.

# ACLs en Linux

Hay un dato que se suele desconocer sin embargo y es que el sistema de ficheros ext2, desde su diseño original, previó la inclusión de este tipo de sistemas de control de acceso y estaban incluídos los enganches (*hooks*) necesarios para su implementación posterior, de forma 100% transparente y compatible hacia atrás.

Pues bien, desde hace varios años existen varios proyectos para incorporar los sistemas basados en ACL en los diferentes sistemas de ficheros soportados por Linux, y

especialmente en ext2 (por ser el tipo de sistema de ficheros más extendido hasta el momento en los sistemas Linux).

Uno de ellos es el proyecto RSBAC, cuyos objetivos son mucho más ambiciosos (realmente mucho más, su objetivo es conseguir un sistema Linux con un nivel de seguridad equivalente al nivel B1 del antiguo Libro Naranja -TCSEC-), pero que incorpora también una implementación 100% operativa de ACLs para ext2.

Otro de ellos es el proyecto Linux Extended Attributes and Access Control Lists, que originalmente tenía como objetivo incorporar el sistema de ACLs al sistema de ficheros ext2 (y posteriormente ext3 cuando este apareció). Posteriormente, al ser el sistema de ACLs elegido por el equipo de Samba para su implementación de ACLs sobre ext2 (para poder ofertar recursos compartidos via SMB con ACLs al igual que los sistemas Windows NT y posteriores) ha sido el candidato oficial de ACLs en ext2 para su inclusión definitiva en el kernel. De hecho, desde la versión 2.5.23 forma parte del kernel estándar.

Para ello se ha hecho un esfuerzo de coordinación y estandarización bastante grande con los desarrolladores de otros sistemas de ficheros como XFS y JFS (principalmente, aunque no sólo con estos) que también soportaban ACLs desde su origen. La idea era ofertar una capa abstracta común en el VFS (*Virtual File System*) de forma que las aplicaciones y el resto del sistema operativo trabajasen por igual, y con la misma API, con cualquiera de los sistemas de ficheros que soportasen ACLs (ext2/ext3, XFS, JFS, ReiserFS, etc.).

El resultado: ya están disponibles los sistemas de ficheros con ACLs en el núcleo estándar, en su versión de desarrollo. Sin embargo, no es necesario esperar hasta la estabilización del actual núcleo de desarrollo para disfrutar de las ventajas de las ACLs. Todos los sistemas de ficheros mencionados arriba están disponibles bien como parte del kernel estándar, bien como parches, para las versiones estables del núcleo. Y son parches con calidad de producción, con lo cual son perfectamente utilizables en entornos cuya estabilidad sea una condición indispensable.

En el caso del sistema de ficheros ext2/ext3, que son el objetivo del proyecto Linux Extended Attributes and Access Control Lists, las ACLs son incluso transparentes para aquellos núcleos que no lleven incorporados los parches necesarios, de forma que si accidentalmente se arranca el sistema con un núcleo sin soporte para ACLs no ocurre absolutamente nada, salvo obviamente que no disponemos de las características avanzadas de las ACLs y sólo tenemos a nuestra disposición el modelo de permisos tradicional.

A condición de que tengamos un ejecutable de e2fsck que soporte ACLs, incluso si el núcleo no lo soporta, podemos ejecutar e2fsck sobre el sistema de ficheros de forma

segura. En caso de tener un e2fsck sin soporte de ACLs, el único problema que tendremos en este caso es la pérdida de las ACLs, pero nunca la pérdida de datos.

Las versiones de e2fsprogs (el paquete donde se incluyen las utilidades de ext2) a partir de la version 1.28 ya incorporan soporte de serie para Atributos Extendidos (Extended Attributes o EAs), que es la característica del sistema de ficheros necesaria para poder implementar las ACLs. En el sitio del propio proyecto se pueden encontrar parches para algunas versiones anteriores de e2fsprogs, en caso de ser necesario.

## Configuración de acceso a ACL

Existen dos tipos de ACLs: *acceso ACLs* y *ACLs predeterminado*. Un acceso a ACLs es la lista de control de acceso para un archivo o directorio específico. Un ACLs predeterminado sólo puede ser asociado con un directorio, si un archivo dentro del directorio no tiene un acceso ACL, utiliza las reglas del ACL predeterminado para el directorio. Los ACLs predeterminado son opcionales.

Los ACLs se pueden configurar:

1. Por usuario
2. Por grupo
3. A través de la máscara de derechos efectivos
4. Para usuarios que no estén en el grupo de usuarios para el archivo

La utilidad `setfacl` configura ACLs para archivos y directorios. Utilice la opción `-m` o modifique el ACL de un archivo o directorio:

```
setfacl -m <rules> <files>
```

Las reglas (`<rules>`) deben ser especificadas en los formatos siguientes. Se pueden especificar múltiples reglas en el mismo comando si estas se encuentran separadas por comas.

`u:<uid>:<perms>`

Configura el acceso ACL para un usuario. Se debe especificar el nombre del usuario o su UID. El usuario puede ser cualquier usuario válido en el sistema.

`g:<gid>:<perms>`

Configura el acceso ACL para un grupo. Se debe especificar el nombre del grupo o su GID. El grupo puede ser cualquier grupo válido en el sistema.

`m:<perms>`

Configura la máscara de derechos efectivos. La máscara es la unión de todos los permisos del grupo propietario y todas las entradas del usuario y grupo.

`o:<perms>`

Configura el acceso ACL para otros usuarios que no esten en el grupo para el archivo.

Se ignoran los espacios en blanco. Los permisos (`<perms>`) deben ser una combinación de caracteres `r`, `w` y `x` para lectura, escritura y ejecución.

Si un archivo o directorio ya tiene una ACL y se usa el comando `setfacl`, se añaden las reglas adicionales al ACL existente o la regla existente es modificada.

Por ejemplo, para otorgar permisos de lectura y escritura para el usuario tfox:

```
setfacl -m u:tfox:rw /project/somefile
```

Para eliminar todos los permisos para un usuario, grupo u otros, utilice la opción `-x` y no especifique ningún permiso:

```
setfacl -x <rules> <files>
```

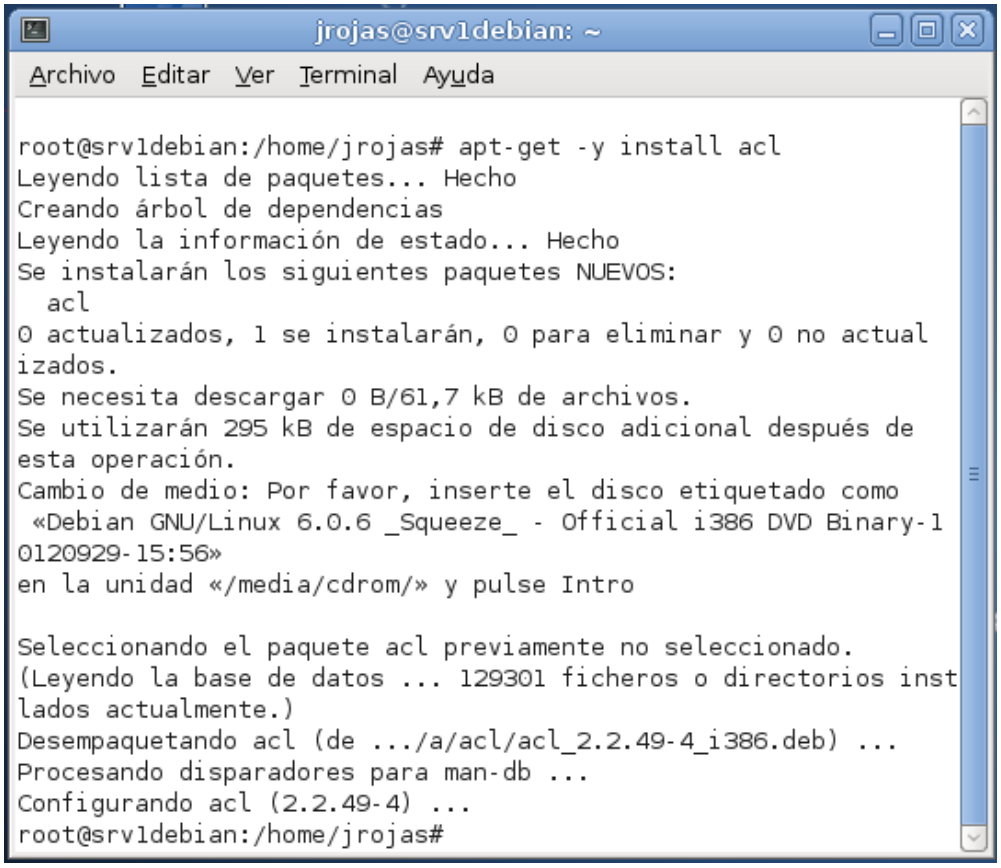
Por ejemplo, para eliminar todos los permisos del usuario con UID 500:

```
setfacl -x u:500 /project/somefile
```

# TALLER ACLs A REALIZAR

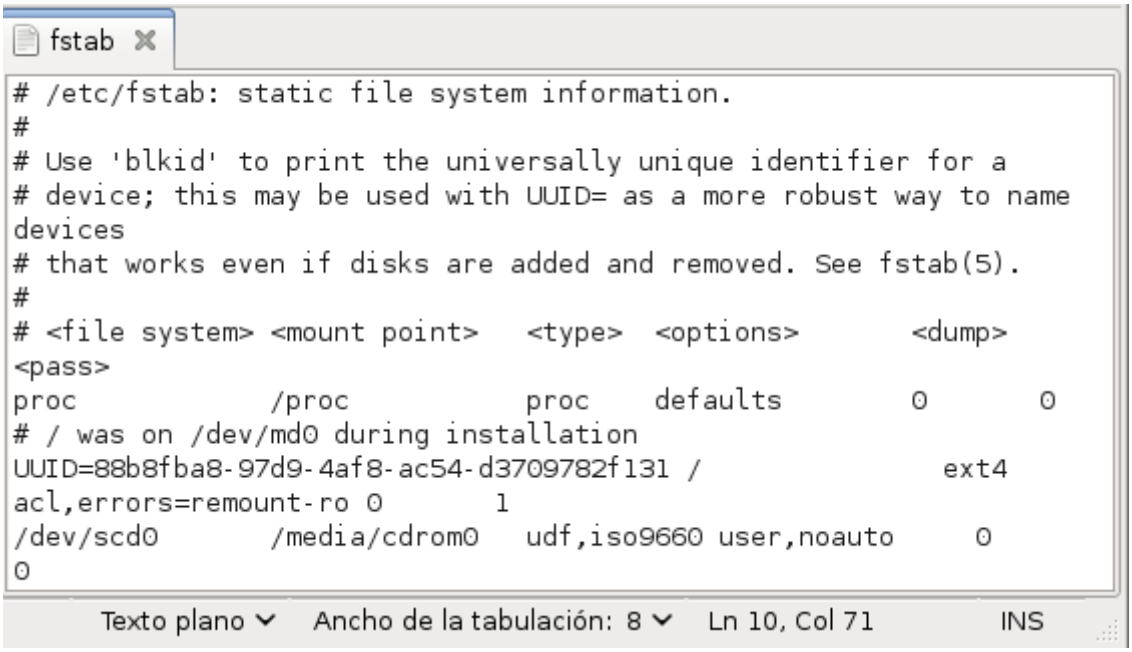
1. **Habilitar el uso de listas de control de acceso en la partición actual de Linux.**

   Instalar paquete:
   **apt-get -y install acl**



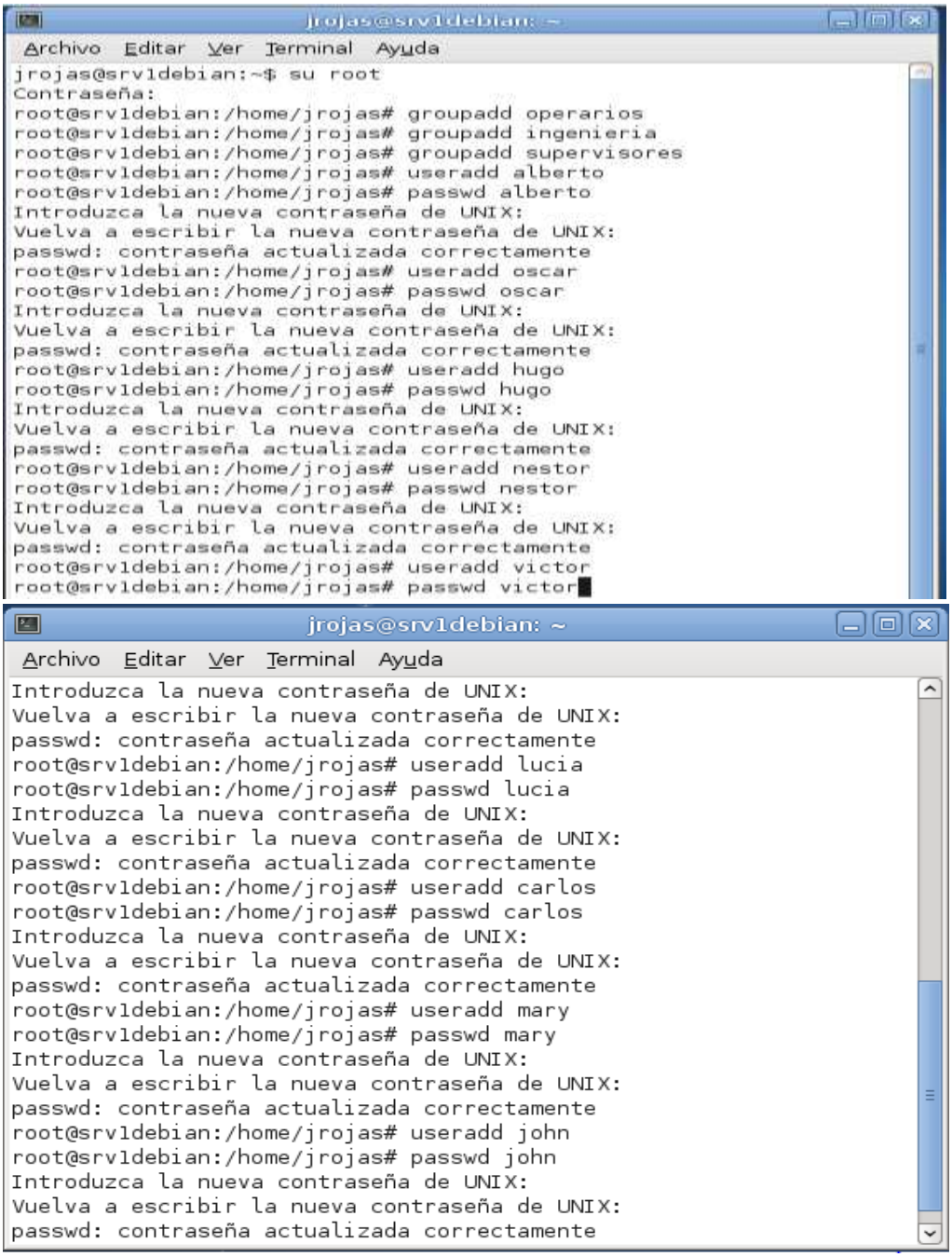

**Permitir su uso en la partición:**

En el archivo **/etc/fstab** adicionamos el permiso acl a la partición donde se van a usar los permisos, de la siguiente forma:

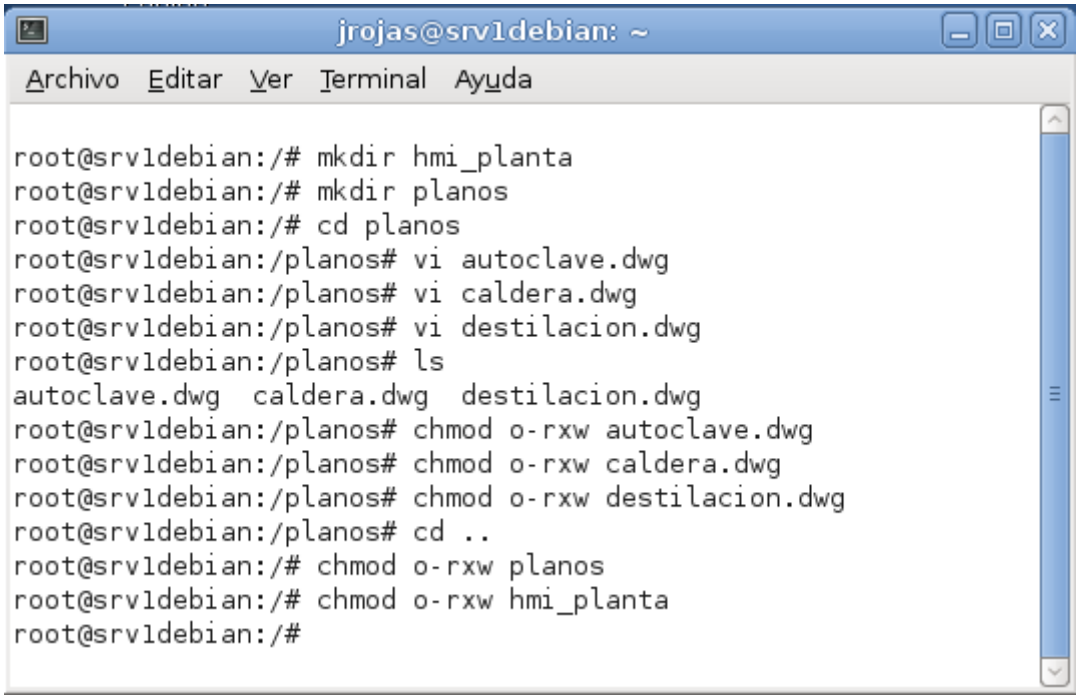



Para que este cambio se realice debemos reiniciar el sistema.

2. **Crear los grupos y usuarios, mediante comandos de Shell, tal como se observa en la figura 1.**

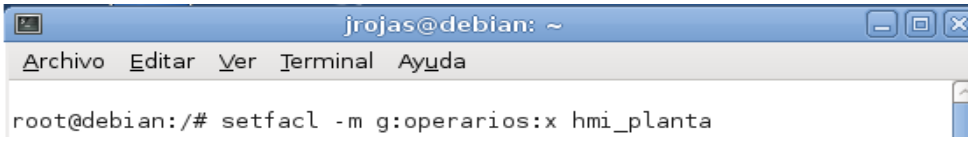



3. **Crear los archivos y directorios mostrados en la figura 1, deshabilitando siempre los permisos de lectura, escritura y ejecución para otros usuarios.**

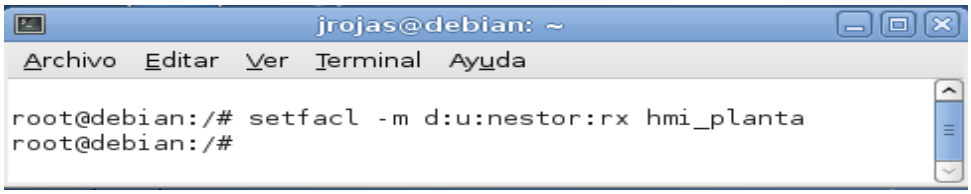



**4. Crear las ACL's necesarias para permitir el acceso a los archivos y directorios, de la manera indicada en la figura 1, con los comandos apropiados de la Shell de Linux.**

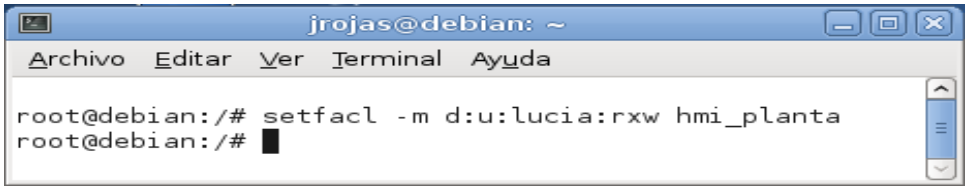
**setfacl -m g:operarios:rwx hmi_planta**



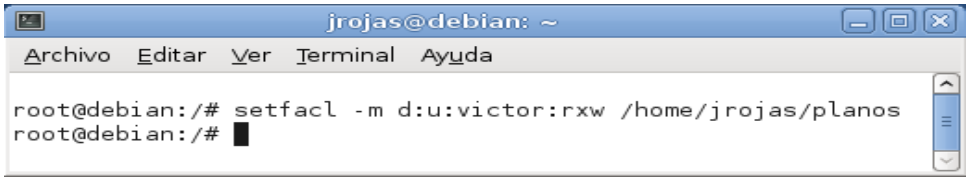

**setfacl -m d:u:nestor:rx hmi_planta**




**setfacl -m d:u:lucia:rxw hmi_planta**




**setfacl -m d:u:victor:rxw /home/jrojas/planos**

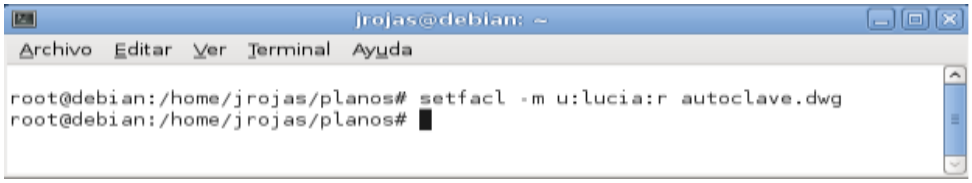



**setfacl -m g:supervisores:rx /home/jrojas/planos**


**setfacl -m u:albert:r /home/jrojas/planos/autoclave.dwg**


**setfacl -m u:oscar:r /home/jrojas/planos/caldera.dwg**


**setfacl -m u:lucia:r autoclave.dwg**



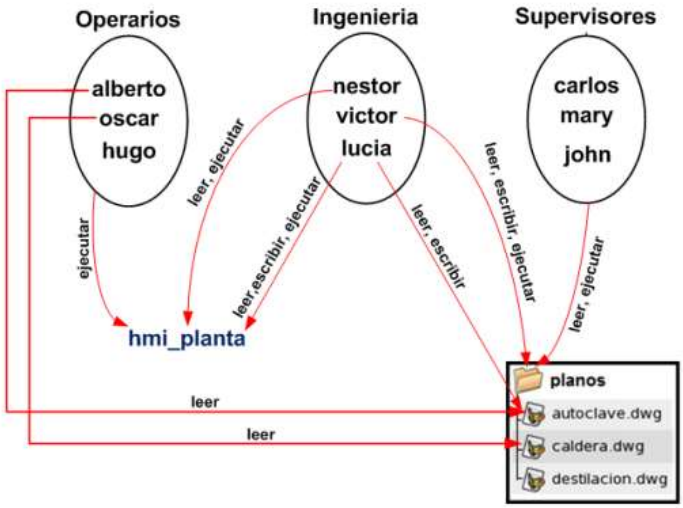
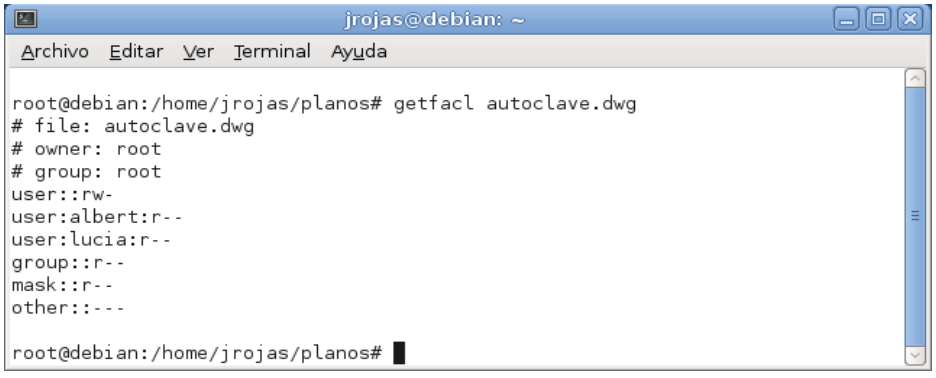



**Figura 1. Esquema de permisos de archivos y directorios mediante ACL's en Linux.**

5. **Efectuar un login, desde diferentes ventanas de Shell, con diferentes cuentas de usuario, para probar la efectividad en la configuración de las ACL's.**


- ACL's de autoclave.dwg: **getfacl autoclave.dwg**

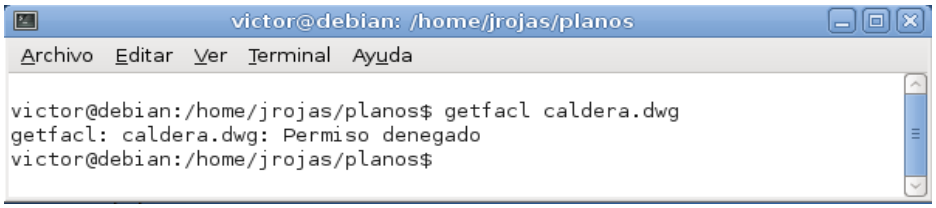
**SESION: Jrojas**




- ACL's de caldera.dwg: **getfacl caldera.dwg**

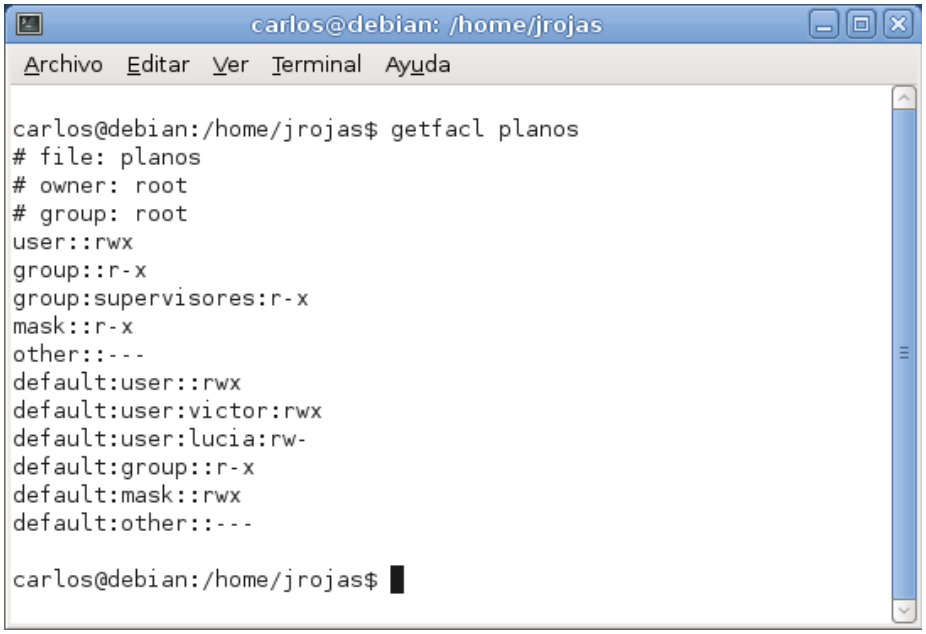
**SESION: Victor**

Como este usuario no tiene permisos sobre este archivo por tal motivo no puede visualizar las acl sobre el archivo caldera.dwg



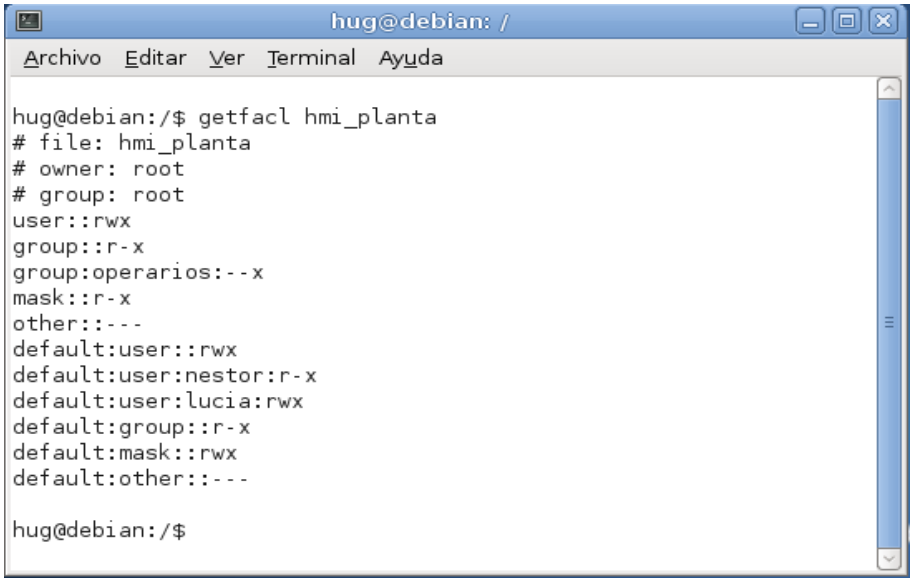

- ACL's de planos: **getfacl planos**

**SESION: Carlos**



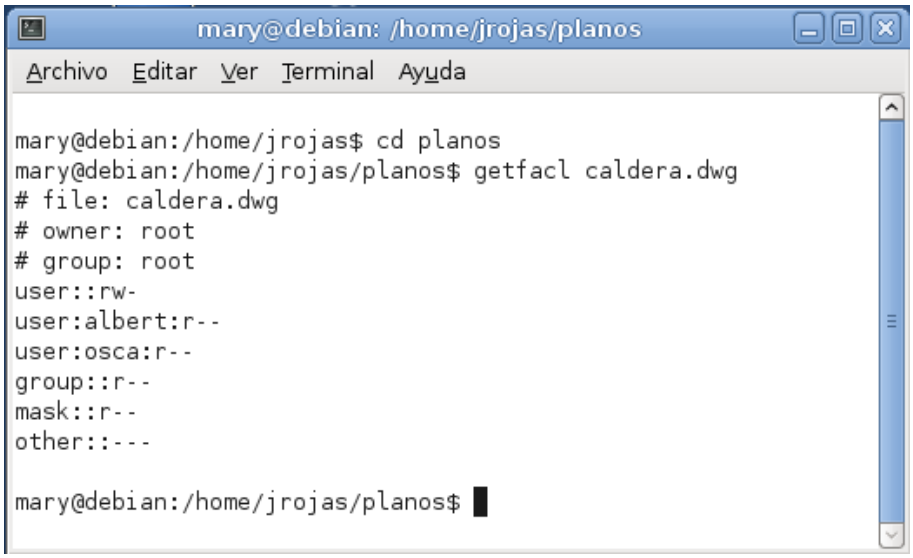

- ACL's de hmi_panta: **getfacl hmi_planta**

**SESION: Hugo**




- ACL's de caldera.dwg: **getfacl caldera.dwg**

**SESION: Mary**




6. **Investigue, con sus compañeros de grupo, qué utilidad tienen los bits SetUID y SetGID de un archivo y cómo se pueden cambiar estos valores en Linux.**

## Los bits *SUID(SetUID)*, *SGID(SetGID)*

Los permisos de los archivos en Unix se corresponden con un número en octal que varía entre 000 y 777; sin embargo, existen unos permisos especiales que hacen variar ese número entre 0000 y 7777; se trata de los bits de permanencia (1000), *SGID* (2000) y *SUID* (4000).

El bit *SUID* o SetUID se activa sobre un fichero añadiéndole 4000 a la representación octal de los permisos del archivo y otorgándole además permiso de ejecución al propietario del mismo; al hacer esto, en lugar de la x en la primera terna de los permisos, aparecerá una s o S sino hemos otorgado el permiso de ejecución correspondiente.

El bit *SUID* activado sobre un fichero indica que todo aquél que ejecute el archivo va a tener durante la ejecución los mismos privilegios de quién lo creó.

Esto también es aplicable al bit *SetGID* pero, en este caso, a nivel de grupos del fichero en lugar de propietario.

Los bits de SetUID y SetGID dan a Unix una gran flexibilidad, pero constituyen al mismo tiempo la mayor fuente de ataques internos al sistema. Cualquier sistema Unix tiene un cierto número de ejecutables setuidados y/o setgiados. Cada uno de ellos, como acabamos de comentar, se ejecuta con los privilegios de quien lo creó --usualmente el root-- lo que directamente implica que cualquier usuario tiene la capacidad en modo privilegiado si es el administrador quien creó los ejecutables. Evidentemente, estas tareas han de estar controladas de una forma exhaustiva, ya que si una de ellas se comporta de forma anormal puede causar daños irreparables al sistema.

Es por esto que conviene estar atentos a los nuevos ficheros de estas características que se localicen en la máquina. Demasiadas aplicaciones de Unix se instalan por defecto con ejecutables setuidados cuando realmente no es necesario por lo que a la hora de instalar nuevo software o actualizar el existente hemos de acordarnos de resetear el bit de los ficheros que no lo necesiten.

Este tipo de ficheros a pesar de presentar ciertos riesgos son estrictamente necesarios en Unix. Un ejemplo clásico es el fichero binpasswd. Este fichero, entre otras, tiene la función de modificar el fichero de claves (etcpasswd o etcshadow). Como no resulta una solución apropiada dar permisos para todos los usuarios al fichero de contraseñas, lo que se hace es activar el bit *SetUID*.