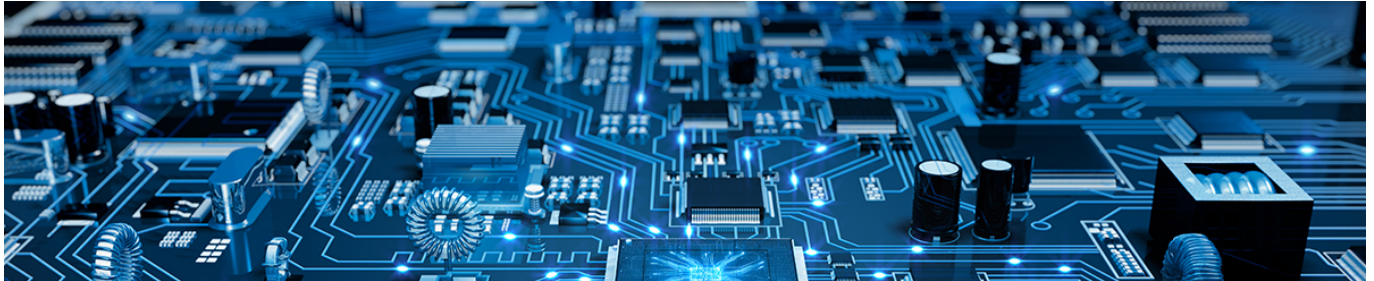


Informe de la práctica 1 de Administración de Sistemas

Escuela Superior de Ingeniería y Tecnología - ULL



Usuarios y recursos en Linux

Autores:

Marcos Padilla Herrera

([alu0101045177](#))

Eric Dürr Sierra

([alu0101027005](#))

El siguiente documento pretende hacer acopio de los distintos aspectos y conclusiones experimentados a lo largo de la primera práctica de Administración de Sistemas. Más adelante encontrará una introducción acerca de la práctica.

Índice

- [Introducción](#)
- [Preparación del sistema](#)
- [Creación de usuarios y contraseñas](#)
- [Directorios y permisos](#)
- [Cuotas de disco](#)
- [Proyectos y ejecutivos](#)
- [Script para la creación de usuarios](#)
- [Referencias](#)



Introducción al informe

Contenido del Informe

En este informe se tratará de mostrar los pasos que hemos seguidos para poder realizar lo que se nos ha propuesto punto por punto

Breve explicación de la práctica

Se nos presenta la siguiente situación de una Organización:

- Existen 3 proyectos: *Aeropuerto*, *Centro Comercial*, *Parque*
- Dos ejecutivos: *ejec1* y *ejec2*
- Seis usuarios: *usu1* *usu2* *usu3* *usu4* *usu5* *usu6*
- La distribución de los usuarios por proyectos es de la siguiente forma:

Usuario	Proyecto		
	Aeropuerto	Centro Comercial	Parque
usu1		♦	
usu2	♦		
usu3	♦	♦	
usu4	♦	♦	
usu5	♦	♦	♦
usu6			♦

- La asociación de ejecutivos a proyectos es:

Usuario	Proyecto		
	Aeropuerto	Centro Comercial	Parque
ejec1	♦		♦
ejec2	♦	♦	

- Los ejecutivos asociados a un determinado proyecto podrán leer la información de ese proyecto, pero no podrán modificarla
- Los ejecutivos que no pertenezcan a un proyecto no deben poder acceder directamente a los directorios de los proyectos

- Para que estos ejecutivos puedan controlar el estado de cada proyecto al que no pertenecen, deben existir en el directorio `/usr/local/bin` tantos programas como proyectos existan.
- Estos programas internamente han de realizar un `"ls"` sobre el directorio del proyecto correspondiente.
- El programa que permite evaluar cada proyecto, debe cumplir lo siguiente debe poder ser ejecutado únicamente por los ejecutivos de la organización y debe tener asignado los permisos suficientes para poder ejecutar el `"ls"` sobre el directorio correspondiente.

Objetivos

- Poner en práctica el conocimiento sobre la creación y manejo de *usuarios* y *grupos*, así como de los *permisos* en ficheros, directorios y ejecutables.
- Utilización de los bits SETUID y SETGID en ficheros ejecutables
- Utilización del bit SETGID en directorios
- Utilización del sticky bit
- Manejo de Listas de Control de Acceso (ACL's)
- Manejo de Cuotas de Disco



Preparación del sistema - configuración de las máquinas remotas

En las siguientes secciones se va a proceder a explicar aquellas acciones tomadas como preparativos para el desarrollo de esta y las siguientes prácticas.

En resumen se trata de la creación de dos máquinas virtuales bajo la estructura IaaS.

IaaS hace referencia a Infrastructure as a Service, más información en [Microsoft Azure](#).



imagen de cabecera de lo que sería un servicio IaaS

Este tipo de servicios permite gestionar de manera instantánea una infraestructura computacional que es gestionada y provisionada directamente desde internet. Evitamos muchos quebraderos de cabeza respecto a cuestiones de hardware y disponibilidad de espacio. Este tipo permite aumentar o decrementar muchas características bajo demanda.

El objetivo de esta parte de la práctica, que fue la primera, era **montar una infraestructura** sobre la cual poder trabajar todos aquellos aspectos a desarrollar durante las prácticas.

La infraestructura se compondría de **dos máquinas**, un **cliente** y un **servidor**. Ambas máquinas debían no solo configurarse con un sistema operativo concreto y unos paquetes necesarios para el desarrollo de la materia, sino que también era preciso configurar una tarjeta de red que vinculase ambas máquinas por una intranet.

Ambos servidores fueron creados bajo el **servicio IaaS** proporcionado por la **ULL** e inicializados con el sistema operativo CentOS mediante un paquete de instalación predefinido. El propósito de usar dicha plantilla es agilizar este apartado y evitar posibles errores que pudieran perjudicar el uso de la máquina o ralentizar la actividad.

Cabe hacer mención a que todos los datos y sistemas que se crean **residen en un CPD** que realmente **existen y son clusters físicos**. Sin embargo *oVirt* hace de **interfaz** y hace transparente toda esa infraestructura para poder centrarnos en el empleo y administración de las máquinas virtuales.

El acceso a estas está protegido de manera que solo se pueda acceder **usando una VPN** o desde la propia red de la ULL

Características comunes a ambas máquinas

Tanto la máquina que actúa de servidor como el cliente presentan similitudes en cuanto a su configuración:

- Ambas usan **CentOS**
- Ambas tendrán un usuario y un root inicialmente

Claramente con contraseñas distintas y personales

- Ambas partes serán configuradas en cuanto a la interconectividad

Se ahonda más adelante sobre este aspecto

Configuración de la máquina virtual

El proceso para iniciar y configurar una máquina virtual mediante el oVirt es tan sencillo como **emplear la interfaz gráfica** para crearla y luego en los menús de selección introducir las opciones.

No vamos a ahondar mucho más en este aspecto ya que está expuesto y documentado en el [campus virtual](#)

Configuración manual de la interfaz de red

Tras haber hecho la instalación de la máquina y configurado algunos aspectos del sistema, toca el turno de **establecer una interfaz de red** que comunique el cliente y el servidor.

En una primera instancia debemos acceder al directorio del sistema operativo que maneja los archivos dedicados a esta labor, osea **/etc/sysconfig/network-scripts**. Es en esta ruta donde tendremos que echar mano de crear el archivo **ifcfg-eth[numero]** que contendrá la información pertinente a un puerto de **comunicación ethernet**.

1. realizamos una *copia* de alguno ya existente. En caso general *ifcfg-eth0*
2. **modificamos y borramos** algunas de las opciones (BOOTPROTO, DEVICE, ...)
3. **añadimos** la opción de dirección IP y NETMASK acorde a nuestro vínculo de comunicación.

El servidor y el cliente se corresponden por una dirección IP similar que los identifica por pares, es decir *192.168.50.numero_par* *192.168.50.numero_impar*.

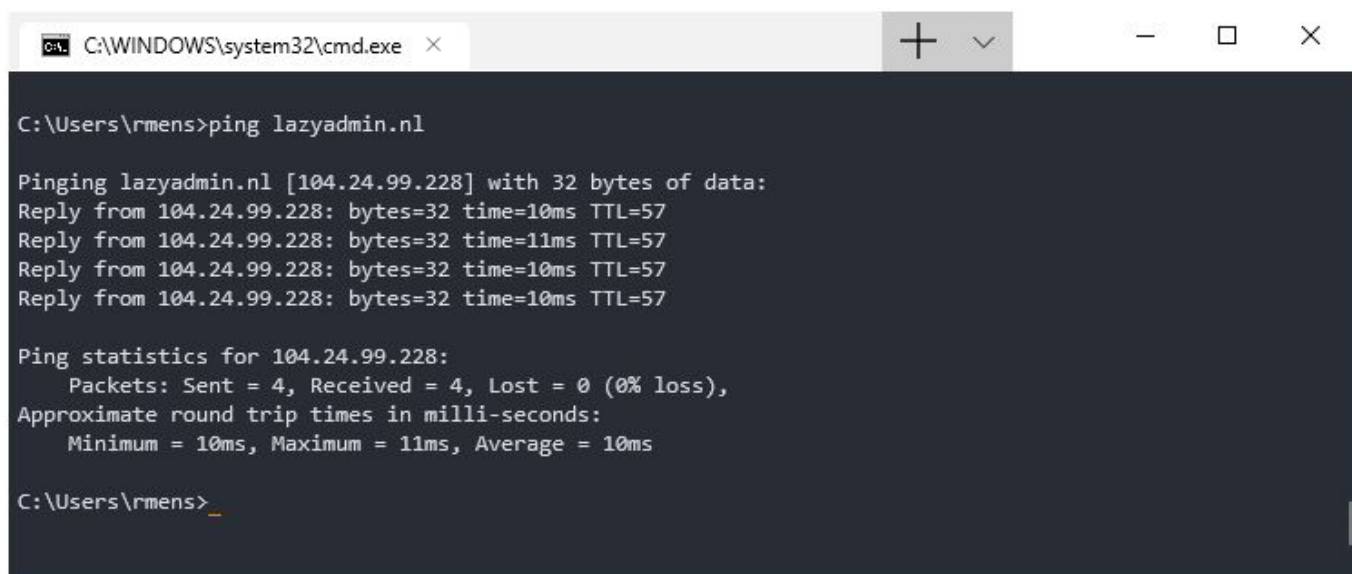
Tras haber hecho esta configuración solamente resta reiniciar el servicio

Para esto hemos usado el comando *systemctl [opción] [servicio]*

Este proceso hay que realizarlo **tanto en el cliente como el servidor**, determinando una dirección IP a cada uno. Se puede comprobar dicho vínculo usando el comando *ping*.

Su uso es: *ping dirección*

De establecerse con éxito una conexión se empezarían a **mostrar paquetes**, tal y como muestra la imagen siguiente:



```
C:\WINDOWS\system32\cmd.exe x
C:\Users\rmens>ping lazyadmin.nl

Pinging lazyadmin.nl [104.24.99.228] with 32 bytes of data:
Reply from 104.24.99.228: bytes=32 time=10ms TTL=57
Reply from 104.24.99.228: bytes=32 time=11ms TTL=57
Reply from 104.24.99.228: bytes=32 time=10ms TTL=57
Reply from 104.24.99.228: bytes=32 time=10ms TTL=57

Ping statistics for 104.24.99.228:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 10ms, Maximum = 11ms, Average = 10ms

C:\Users\rmens>
```



Administración de usuarios y contraseñas

En primer lugar, crearemos un conjunto de usuarios usu1, usu2, usu3, usu4, usu5, usu6, que deben cambiar cada 3 meses su contraseña, que coincidirá con el nombre de usuario. Además, será necesario notificar a los usuarios 1 día antes de que caduque su contraseña, y 2 días después de su caducidad, la cuenta quedará desactivada.

Será necesaria la elaboración de 2 pasos previos que servirán para proceder a la realización de este primer apartado de la práctica:

Paso 1.

Creación de usuarios.

Para la creación de usuarios tenemos que ejecutar el siguiente comando como *superusuario (root)*:

useradd [nombre del usuario] Este comando será necesario para añadir todos los usuarios. Cabe destacar que todos los directorios personales de los usuarios son creados en el directorio **/home** automáticamente con el mismo nombre que el propio usuario. Una vez creado los usuarios, se procede con el siguiente paso.

Paso 2.

Creación y administración de contraseñas

Para asignar una contraseña a cada usuario, utilizaremos el comando:

passwd [usuario a modificar] En nuestro caso queremos que las contraseñas sean iguales al nombre del usuario. Una vez realizado este paso, deberemos modificar las características de la contraseña.

Características de las contraseñas

Para realizar este proceso es necesario modificar el archivo **/etc/shadow**, mediante el comando **chage**:

Se podría abrir el fichero mediante un editor de texto convencional pero debido a la importancia del texto y su fragilidad, sería mucho mas conveniente y seguro trabajar con un comando que se dedicara explícitamente a la modificación del fichero **/etc/shadow**

```
chage --maxdays 90
```

```
chage --warndays 1
```

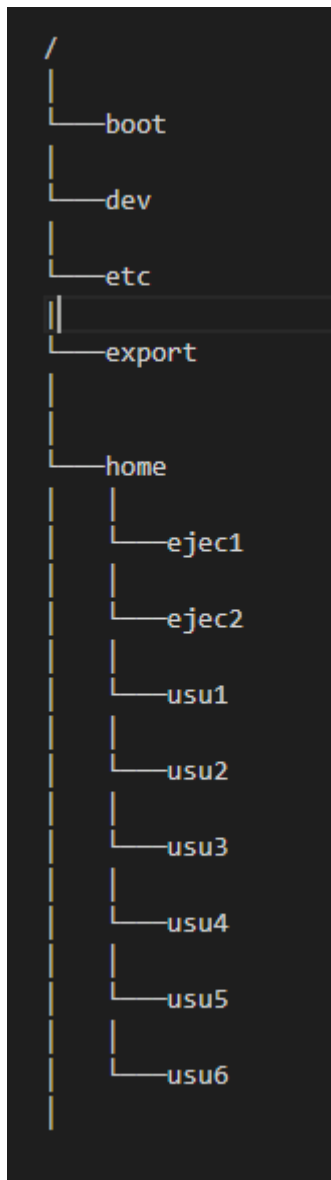


```
chage --inactive 2 [ usuario ]
```

Con la opción **maxdays** modificamos el tiempo máximo que puede transcurrir con la contraseña actual. Una vez sobrepasado este tiempo es obligatorio su cambio. En cuanto a **warndays**, su uso es necesario para indicar que el usuario será notificado un día antes (en este caso) de la caducidad de su contraseña. Por último, la opción **inactive** indica los días que habrá entre la caducidad de la contraseña y el bloqueo de la cuenta del usuario.

```
usu1:$6$b5wyAGwz$CR6Hk.q7XDAW7QYc/nx8B6VkfPsJufbr7vZ04KZM.uawb6vx6gV/D3LG7A3BfouWIH7h2Gdv.WRTTBd8C.TZ0:18311:0:90:1:2::
usu2:$6$mqReEZiT$SDyKoihPXAsTfwTdRplLDpBJ00ynaul6Hk3toS9qEs3bEgt3q7AY0.CvtGGtp2F4UbC1NzNR3Itu5LJLbGaX01:18311:0:90:1:2::
usu3:$6$nm0PlgNi$dndotHLbaL63jiXjDHemY7iU7pg8gw28K..7IFv7KbxKM5CiIjBBPeBbcKk8qMwByH/ZW8xxFCmebTeu05VSq.:18311:0:90:1:2::
usu4:$6$U6pHfYsU$e.DNEVqQbeClESY4Tdxp9c6AcblZm/rRJ1n4DwZMXXn23lNwFM9di1gA5YIxw0BcQZVvOuKxSx9t9X4Muugl0:18311:0:90:1:2::
usu5:$6$myb1qUqP$fVLMvP49XsCZxUdTRmRjkLGbQq76qT0oWGuFaQah5F4.h5He8d/mv1KGi1sRw4dXoL3N/RfBkCkTjGLV09sth.:18311:0:90:1:2::
usu6:$6$i/p2MUgC$OGGcN6kQb5dl1eoJQ1NNZpF/tCiZFvEb.EhsIDE1a.gOfB1Kt0bctBnoXonJfwHcgQzf4.Hc8oZ.B/tJ62N5T0:18311:0:90:1:2::
ejec1:$6$qyeINoGc$OH0XxvKCVuWR78o0qrb0R3eLwAUbr0F0igDpv.6kqgdUTHfLKwBwjDzlu9AEdGF1Zehn2If2vhXV3Yt5cj2vD0:18311:0:90:1:2::
ejec2:$6$Hmv04gxd$Xkcr2C4/shdpo/DFSG0GLe9K0k.Csgp7M5TrH8MHZ4Y8PuY.um.b4IMVHf/Mhn8NxJG4VmAkjt09npSSvnp6k0:18311:0:90:1:2::
```

Acontinuación mostramos el árbol de directorios



Acciones tomadas para el manejo de directorios y permisos de los usuarios del sistema

Ahora vamos a analizar cómo se ha manejado los directorios personales de cada usuario y los permisos que se le asignan. Todo usuario del sistema debe poseer un subdirectorio del propio directorio `__home__` cuyo nombre debe coincidir con el de la cuenta del usuario. Sin embargo, no es necesaria la creación de este directorio personal del usuario, puesto que se crea automáticamente con la creación de usuario.

Además, ningún usuario puede entrar en el directorio personal de otro usuario; sin embargo, aunque un usuario puede crear y borrar todo tipo de archivos en su propio directorio personal, no podrá cambiar los permisos de ese directorio.

Permisos del directorio personal

Para que ningún usuario pueda acceder al directorio personal ni a los ficheros de otro usuario, lo primero que debemos hacer es cambiar los permisos de cada directorio personal. Para ello, utilizaremos el comando `__chmod__` seguido de los permisos que se quieran permitir y el nombre del directorio:

```
chmod 770 usu1 chmod a+rwX,o-rwx usu1
```

Lo que hacemos con este comando, estamos indicando mediante la cifra **770** lo mismo que se puede hacer con la opción **a+rwX,o-rwx**, que el directorio `usu1` tendrá permisos de lectura, escritura y ejecución para el propietario del directorio y para el grupo propietario. Asimismo, cualquier otro usuario no tendrá ningún permiso sobre ese directorio. Además, queremos que el usuario no pueda modificar estos permisos que hemos asignado; el problema es que el propietario de un directorio siempre va a poder cambiar los permisos de ese directorio, así que lo que haremos será hacer propietario a **root** mediante el comando **chown**, y como grupo propietario será el propio **usu1**, éste tendrá la posibilidad de entrar en su directorio

```
chown root usu1
```



Asignación de cuotas de discos

Lo que se demandaba era que cada usuario tuviera una cuota de disco de `__50Mb__` que no podrá sobrepasar `__temporalmente (2 días)__`.

Para realizar este proceso, lo primero que se debe hacer es abrir el archivo `__/etc/fstab__` con un editor de texto, se nos mostrará las distintas particiones del equipo, en nuestro caso nos interesa la cuota de disco para `__/home__`, por lo que añadiremos la opción `__usrquota__` y realizamos los siguientes comandos:

```
mount -o remount /home quotacheck -cug /home quotacheck -avug
```

Con esta serie de comandos lo que hacemos es lo siguiente, primero montamos el archivo, luego se crea el archivo **aquota.user**, se crea la tabla de uso de disco y a partir de aquí habrá que modificar el archivo para establecer las cuotas de disco. Para ésto último paso, se hace uso del comando **edquota usu1** que abre un fichero en el que debemos modificar la columna **soft** y añadirle una cuota de *50Mb*

//insertar foto edquota usu1//

Una vez que hayamos hecho este proceso para cada usuario, el comando **edquota -t** nos abre un fichero en el que cambiaremos el periodo de gracia a *2 días*.

Para comprobar que se ha realizado correctamente las operaciones de las cuotas de disco podemos usar el comando: **repquota /home**

//insertar foto repquota /home//



Proyectos y ejecutivos - cómo se organiza la administración

A continuación se pretende exponer las cuestiones que conciernen a los preparativos para el desarrollo de la actividad laboral en los distintos proyectos para la administración que va a usar el sistema. De este modo tendremos que atender a que cada directorio disponga de un control de permisos para cada uno de los usuarios y ejecutivos que concentre cada proyecto.

Los aspectos serán expuestos uno a uno a continuación

Modelo organizacional

La organización necesita un entorno en el que trabajar. Para ello se crea una estructura de directorios que desde una carpeta de proyectos va dividiendo los archivos pertinentes en la carpeta que corresponde al proyecto.

Esta estructura no es de acceso libre para todos los usuarios ni para los ejecutivos. Estos deben ser distribuidos entre los proyectos pertinentes a cada uno. En resumen solo los usuarios que trabajen en un determinado proyecto podrán leer, escribir, modificar y crear archivos. Siempre y cuando estos archivos sean creados por ellos, en caso contrario solo tendrán permisos de lectura al igual que el ejecutivo de dicho proyecto. Este no podrá más que leer los archivos siempre y cuando supervise el proyecto. En caso contrario no tendrá ningún permiso

La siguiente tabla ilustra la división de responsabilidades que define el modelo:

Usuario	Proyectos		
	Aeropuerto	CentroComercial	Parque
usu1		.	
usu2	.		
usu3	.	.	
usu4	.	.	
usu5	.	.	.
usu6			
ejec1	.		.
ejec2	.	.	

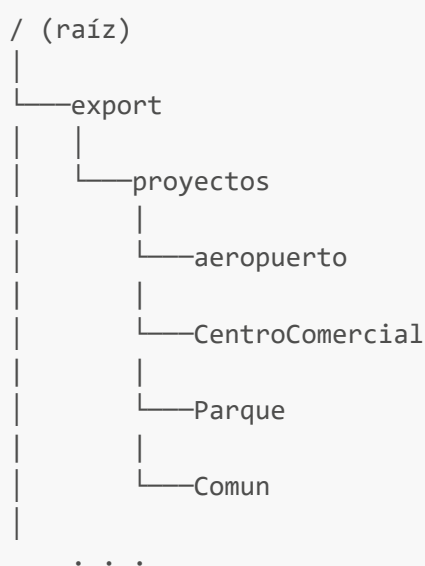
El punto "." marca que el usuario ha sido asignado al proyecto

La principal diferencia, que no es ilustrada, entre ejecutivos y usuarios es el tipo de acceso. Los usuarios, en sus proyectos, van a requerir de permisos de lectura y escritura. Sin embargo los ejecutivos, en los proyectos asignados, solo podrán entrar y leer. De no serles asignados solo podrán revisar el contenido del directorio de cada proyecto.

Más adelante se explica como se ha procedido para permitir hacer estas operaciones

La estructura de directorios que engloba esta organización se compone por una carpeta por cada proyecto más una denominada *comun* a la cual podrá acceder cualquier usuario del sistema. Con la salvedad de que en esta última sólo el usuario que crea el archivo podrá borrar sus propios archivos, de modo que todos podrán modificar y crear cualquier archivo

La estructura del árbol de directorios es la siguiente:



Cómo se han configurado los directorios

Dado que buscamos que solo los usuarios asignados a un proyecto concreto puedan realizar las operaciones necesarias; la forma de que podamos controlar los permisos de una manera sencilla es haciendo que el propietario sea root (como ya es norma si no queremos que se nos perjudique en este sentido) y proporcionando, solo al grupo propietario los permisos pertinentes.

Acto seguido lo que debemos hacer es asignar los usuarios que van a trabajar en un proyecto al grupo que lo concierne.

La lista de permisos quedaría de la siguiente forma:

```
d---rws---+ root [grupo_de_proyecto] ... [proyecto]
```

y para *comun*:

```
d---rws--T+ root [grupo_de_comun] ... comun
```

Los aspectos a tener en consideración son:

- los permisos "especiales" van a actuar acorde con los directorios (no con ejecutables)
- El uso de setGID para que el grupo propietario del fichero creado sea el del directorio que lo contiene.

de esta manera conseguimos que se puedan modificar los archivos entre usuarios solo del mismo proyecto.

- Que setGID esté en minúscula implica que el permiso de ejecución está concedido también.

tal y como se precisa para poder acceder al directorio.

- An común se ha añadido un sticky, que garantiza que solo el creador del fichero pueda borrar sus archivos.
- Al contrario que con el permiso anterior la " mayúscula indica la ausencia del permiso de ejecución.

no nos interesa que cualquier usuario tenga permisos de ejecución del directorio.

- La aparición del signo "+" indica que una ACL también está presente

La ACL o Access Controll List por sus siglas en inglés va a llevar la labor de permitir extender las limitaciones de los permisos UGO. De esta manera podremos definir de una manera más concreta unos permisos específicos para otros grupos (como pudiera ser el de los ejecutivos de cada proyecto)

Los ejecutivos en nuestro proyecto

De cara a la maniobrabilidad de los permisos y por si la empresa precisara de la inclusión de más ejecutivos para un mismo proyecto; el paso correcto a dar es la creación de un grupo bajo el que reunir a los ejecutivos responsables del proyecto que los reune.

Sencillamente estos grupos podrán ser incluidos a la lista de permisos de la ACL de cada proyecto mediante la sentencia:

```
setfacl -m [u/g]:[nombre]:[permisos]: [ruta]
```

de esta manera podemos indicar por línea de comandos y para cada proyecto; que grupos de ejecutivos y que permisos poseen. un ejemplo podría ser:

```
setfacl -m g:ejecutivos_aeropuerto:rx: /export/proyectos/aeropuerto
```

Con la anterior sentencia hemos modificado la ACL del directorio aeropuerto de manera que aquellos miembros del grupo *ejecutivos_aeropuerto* tiene permisos de lectura y ejecución del directorio aeropuerto, lo cual les permite entrar y revisar su contenido.

Todos los cambios pueden verificarse con la lectura de la ACL mediante el comando: *getfacl [ruta]*

Cabe destacar que estas órdenes las podremos dar solamente como *root* o con sus permisos.

Cuando un ejecutivo no es responsable de un proyecto no podrá entrar en él. Sin embargo deberá de poder revisar su contenido.

Para poder llevar a cabo este aspecto la solución propuesta consiste en la copia y modificación del comando *ls* siendo este ligeramente modificado en cuanto a permisos.

La finalidad de modificar el *ls* es poder acceder bajo el grupo de acceso de ejecutivos pero con limitaciones. Dado que es un ejecutable, al asignarle el permiso setGID tendrá un comportamiento ligeramente distinto. De esta manera se compartiran de manera temporal algunos permisos que no podría obtener de otra manera.

El comando modificado tendría el siguiente aspecto:

```
---r-s---+ root [grupo_proyecto] ... ls-[proyecto]
```

De esta manera, el ejecutivo que lance el programa tomará temporalmente el permiso de lectura que necesita del grupo de proyecto adecuado. Tras esto se le mostrará la lista del contenido, tal y como lo haría *ls*.

Recordemos que el permiso de ejecución se muestra con la "s" minúscula

Con esto ya podríamos considerar una correcta configuración de los proyectos y su entorno de trabajo.

![[logo]](icono-ull-negro.png)

Script para la creación de usuarios

En el siguiente apartado se pretende exponer cómo se ha automatizado el proceso de adición de un nuevo usuario a nuestro sistema.

Por medio del script "[usuarios.sh](#)"



Incluir un usuario conlleva una serie de procesos tales como la modificación de algunos permisos, credenciales de contraseña y demás características que deben hacerse por cada uno de los nuevos participantes.

Este proceso puede resultar, a parte de tedioso, un **momento en el que pueden cometerse errores**.

Es entonces donde escribir un *script* que automatice el proceso y que revise aquellas tareas que no requieran de intervención personal aporta una solución y una mejora en cuanto a la eficiencia del sistema.

En el programa que hemos creado empleando **bash** cuidamos que:

1. Se cree el usuario con un nombre indicado al llamar el programa
2. Las contraseñas sean semejantes al nombre de usuario
3. Se cree la carpeta personal del usuario con los debidos permisos

Se puede observar como el programa realiza las mismas acciones que se cometen durante el proceso del guión de la práctica, tal y como se ha ido indicando en el informe pero esta vez en una sola orden.

El siguiente paso va a ser **analizar** paso por paso como funciona el *script*.

Primero es crucial **comprobar que el usuario que ejecuta el script es root** para ello nos servimos de los condicionales y del valor por expansión `$USER` que revelará el nombre del usuario actual.

```
if [ $USER == "root" ]; then
    ...
else
    echo "Denied, not enough permissions"
fi
```

Una vez verificado que la identidad del usuario que lanza la orden es **el administrador** (o que al menos actúa como tal), se procede a la **comprobación** de que se ha proporcionado un **nombre de usuario** para poder llevar a cabo la orden. Nuevamente haciendo uso de los condicionales.

```
if [ $# -lt 1 ]; then
    echo "---Debe proporcionar un nombre de usuario para poder comenzar---"
    echo "Ejemplo: "$0 "my_username"
    exit
fi
```

En este caso, tal y como se observa en el código expuesto, se realiza una comprobación gracias a la expansión del número de argumentos dada por `$#`.

De ser esta menor que 1 se procederá a imprimir un mensaje de ayuda y a salir del programa.

`$0` corresponde a la orden leída por bash, que es el nombre del programa.

Tras la verificación de estos aspectos esenciales se prosigue a **dar valores a las variables** y **comprobar su duplicidad**. Ya que de existir en el sistema un usuario con el nombre proporcionado **el programa acabaría en error**. Para comprobar esto se emplea una sentencia de bash que devuelve la existencia del usuario (*getent*) en el fichero "passwd". En caso de no ser nula se proporciona un mensaje de error y se procede a terminar el programa de inmediato.

Las siguientes líneas de código corresponden a lo explicado:

```
newuser=$1
newpasswd=$1

if getent passwd $newuser > /dev/null 2>&1; then
    echo "user already exists"
    exit
fi
```

`$1` se expande con el primer argumento proporcionado, que debería ser el nombre de usuario.

Finalmente, lo que resta es ejecutar la serie de **comandos que registran a nuestro usuario** y le proporcionan una carpeta personal.

```
useradd $newuser
echo $newuser:$newpasswd | chpasswd
echo "Added user: $newuser"
echo "User's password for $newuser is $newpasswd. Change it as soon as
possible."
echo "----- creating home directory -----"
mkdir /home/$newuser
echo "---- changed permissions ----"
chgrp $newuser /home/$newuser
chmod u+rx /home/$newuser
chmod g+rx /home/$newuser
chmod o-rwx /home/$newuser
```

Se puede apreciar que la contraseña proporcionada es el mismo nombre dado para el usuario .Sin embargo se advierte de cambiarla cuanto antes.

Este *script* es, en muchos aspectos, mejorable. Por ejemplo:

- modificando también aspectos tales como las opciones en */etc/shadow*
- generando una contraseña aleatoria que se le suministre solo al usuario
- permitiendo más argumentos para la adición a grupos de trabajo

Sin embargo por cuestiones de falta de tiempo no se ha ahondado en estos aspectos de cara al día de la entrega.



Apartado de referencias usadas para el desarrollo de la práctica y el informe

- [Enlace a RedHat sobre cuota de disco](#)
- [Enlace sobre la administración de usuarios](#)
- [Enlace a RedHat sobre permisos en ficheros y directorios](#)
- [Enlace sobre el uso del comando chage](#)
- [Enlace sobre las ACLs](#)
- [Enlace de interés sobre CentOS](#)
- [Enlace sobre permisos especiales](#)

