LINUX - El intérprete de órdenes I

Objetivo:

Se pretende introducir al alumno en el uso del intérprete de órdenes *bash* de LINUX.

El intérprete de órdenes

Como todos los sistemas operativos de tipo UNIX, LINUX dispone de un intérprete de órdenes (en inglés se utiliza la palabra *shell*) que hace de interfaz entre el usuario y el propio sistema operativo y cuyo nombre es ***bash*** (acrónimo de ***B**ourne **A**gain **SH**ell*). En condiciones normales, el intérprete de órdenes muestra en pantalla un indicador de línea de órdenes (en inglés se utiliza la palabra *prompt*) esperando que el usuario introduzca una orden. Una vez introducida, el intérprete de órdenes la analiza y si no encuentra ningún error, la ejecuta cediendo el control al sistema operativo. Al finalizar su ejecución, el sistema operativo devuelve el control al intérprete de órdenes que muestra de nuevo el indicador de línea de órdenes para que el usuario pueda seguir introduciendo más órdenes.

Orden simple

Una orden simple es una secuencia de palabras separadas por espacios y redirecciones y que terminan con un operador de control (normalmente un retorno de carro). La primera palabra especifica la orden a ser ejecutada y las palabras restantes se pasan como opciones y argumentos de la orden.

Sintaxis: **orden [** *opciones* **] [** *argumentos* **]**

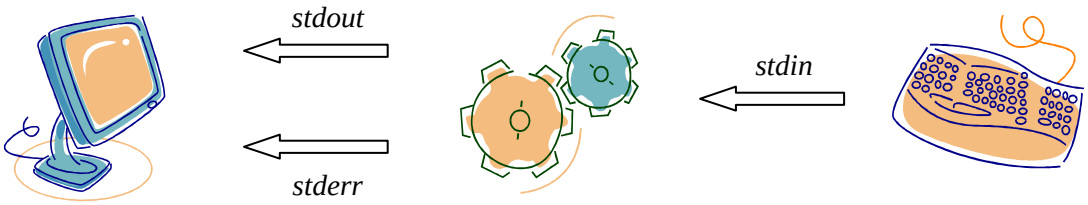
> *Orden:* ***echo [opciones] [cadenas]***
> Esta orden muestra por pantalla las cadenas de texto separándolas con un espacio.
>
> *Ejecución:* `echo –n "Hola mundo"`
> Muestra `Hola mundo` por pantalla. Por defecto, la orden `echo`, después de mostrar todas las cadenas, introduce un salto de línea. Con la opción `–n`, esta orden elimina ese carácter de salto de final de línea.
>
> *Orden:* ***man [opciones] [orden]***
> Esta orden muestra por pantalla información sobre otras órdenes (es la ayuda del *bash*).
>
> *Ejecución:* `man echo`
> Muestra por pantalla información sobre la orden `echo`. Para salir de la ayuda pulsar la letra '`q`'. Para la búsqueda de una palabra '`/palabra_a_buscar`'.

Por defecto, la información presentada por la orden `man` consta en primer lugar de un pequeño resumen sobre el uso más común de la orden sobre la que se pide información, seguido de una sinopsis sobre las distintas formas de utilizarla y finalmente una descripción detallada para cada una de sus opciones. Se **RECOMIENDA** utilizar la orden `man` para todas las órdenes que se presentan a lo largo de este guión.

> **Ejercicio:**
> 1.- Visualizar la ayuda sobre las órdenes `cat` y `find` en una única orden.

Entrada, salida y error estándar

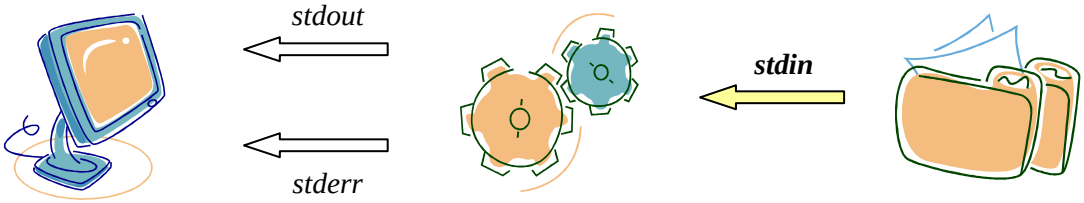
Cada vez que ejecutamos una orden, el sistema operativo le abre automáticamente tres interfaces (en los sistemas operativos tipo UNIX se utiliza el término fichero): la *entrada estándar*, la *salida estándar* y el *error estándar*.
La entrada estándar (*stdin*) se refiere al fichero por el que una orden recibe su entrada (por defecto, es el teclado).
La salida estándar (*stdout*) se refiere al fichero por el que una orden presenta sus resultados (por defecto, es la pantalla o más concretamente la ventana en la que se está ejecutando el intérprete de órdenes).
El error estándar (*stderr*) se refiere al fichero por el que una orden presenta los mensajes que va generando cuando ocurre un error (por defecto, también es la pantalla).



Antes de que se ejecute una orden, es posible redirigir cualquiera de sus ficheros. A esta acción se le conoce como ***redirección***. Para llevarla a cabo es necesario utilizar los ***operadores de redirección*** que se procesan en el orden en el que aparecen.

Redirección de la entrada estándar

Cuando se quiere redirigir la entrada estándar de una orden a un archivo, es necesario utilizar el operador de redirección '<' seguido del nombre del archivo. En este caso, una orden lee los datos de entrada que necesita desde el archivo señalado.



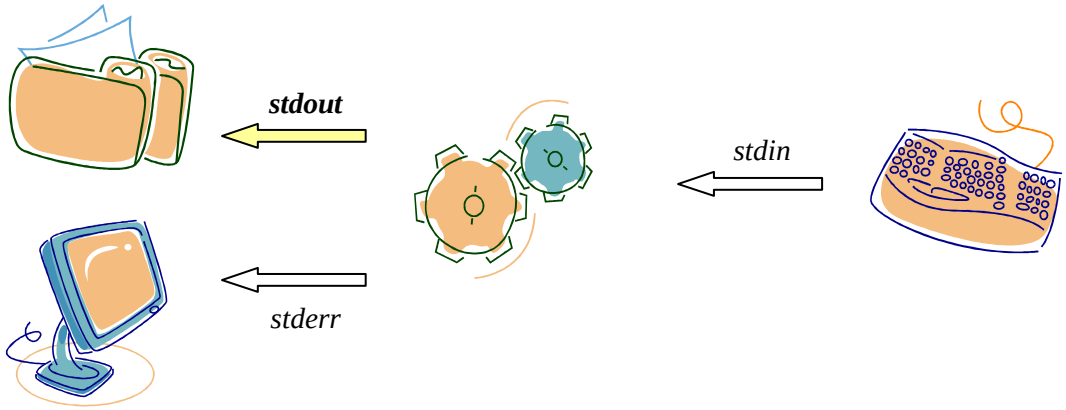
Orden: ***cat [opciones] [archivos]***
Esta orden muestra por pantalla el contenido de los archivos por la salida estándar. Si no se especifica ningún archivo, la orden repite todo lo que se ha escrito por la entrada estándar. NOTA: para indicar que se ha acabado de escribir hay que pulsar CTRL+D.

Ejecución: **`cat .bashrc`**
Muestra por pantalla el contenido del archivo `.bashrc`.

Ejecución: **`cat < .bashrc`**
El resultado que se obtiene es idéntico al ejemplo anterior, salvo que en este caso la forma de conseguirlo es diferente. La orden `cat`, esta vez, se ejecuta sin argumentos por lo que la orden espera que los datos lleguen por la entrada estándar. Pero al redirigir la entrada estándar con un archivo, el intérprete de órdenes lee el archivo y le va pasando los caracteres como si estuviésemos escribiéndolos.

Redirección de la salida estándar

Cuando se quiere redirigir la salida estándar de una orden a un archivo, es necesario utilizar el operador de redirección '`>`' seguido del nombre del archivo. En el caso de no existir, el archivo se crea. En el caso de existir, el archivo se vacía antes de hacer la redirección. En el caso de que se quiera añadir la salida estándar de una orden sin borrar el contenido de un archivo que ya existe, el operador de redirección a utilizar debe ser '`>>`'.



Ejecución: **`echo "Hola mundo" > prueba.txt`**
Suponiendo que el archivo `prueba.txt` no existe, se crea dicho archivo y se guarda en él la cadena `Hola mundo`.

Ejecución: **`cat > prueba2.txt`**
Si se redirige la salida estándar de la orden `cat` a un archivo, todo lo que se escriba por la entrada estándar (el teclado) se almacena en dicho archivo. Esta es una forma fácil y rápida de crear archivos de texto.

Orden: ***ls [opciones] [ficheros]***
Esta orden da un listado del contenido de directorios.

Ejecución: **`ls`**
Muestra el contenido del directorio actual.

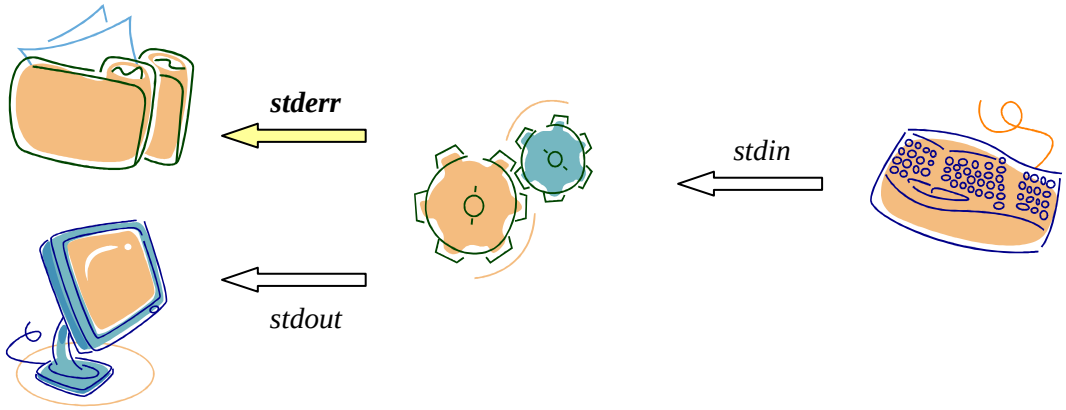
Ejercicio:
2.- Redireccionar la salida estándar de la orden `ls -al` al archivo `listado.txt`. Ver el contenido del archivo con la orden `cat` y explicar que se ve.
(NOTA: La opción `-al` consta en realidad de 2 opciones `-a` y `-l`. En los casos en los que las opciones se representan con una única letra, el intérprete de órdenes permite juntarlas detrás de un único guión).

Redirección del error estándar

Cuando se quiere redirigir el error estándar de una orden a un archivo, es necesario utilizar el operador de redirección '`2>`' seguido del nombre del archivo. Como en el caso anterior, en el caso de no existir, el archivo se crea y en el caso de existir, el archivo se vacía antes de hacer la redirección. En el caso de que se quiera añadir el error estándar de una orden sin borrar el contenido de un archivo que ya existe, el operador de redirección a utilizar debe ser '`2>>`'.

Orden: ***find [camino(s)] [condición(es) de búsqueda] [acciones]***
Esta orden permite buscar archivos recorriendo árboles de directorios especificados por los caminos, evaluando de izquierda a derecha las condiciones de búsqueda. Además se pueden especificar acciones sobre los resultados obtenidos.

Ejecución: **`find /etc –name "a*" –size +1024k 2>/dev/null`**
Búsqueda de aquellos archivos que empiezan por `a` mayores de 1 Mbyte en el directorio `/etc` y en sus subdirectorios. Se redirige la salida estándar de error al dispositivo nulo (Probad esta orden sin la redirección del error estándar).



La orden `find` es muy completa. Utilizando operaciones lógicas podemos concatenar varias condiciones que nos permitan realizar búsquedas mucho más complejas y exactas. Para más información consultar el manual de la orden (`man find`)

Ejercicios:
3.- Utilizar la orden `find` para encontrar **en el directorio `/etc` y en sus subdirectorios** aquellos archivos que empiezan por "`sa`", que tengan un tamaño inferior a los 2 kbytes y redirigir el error estándar a un archivo denominado `fallos.txt`. Explicar que se ve en ese archivo.
4.- Volver a hacer la búsqueda anterior limitándola **solo al directorio `/etc`** redirigiendo esta vez el error estándar a un archivo denominado `fallos2.txt`. Explicar en que se diferencian los resultados entre el ejercicio anterior y éste.

El sistema de archivos en LINUX

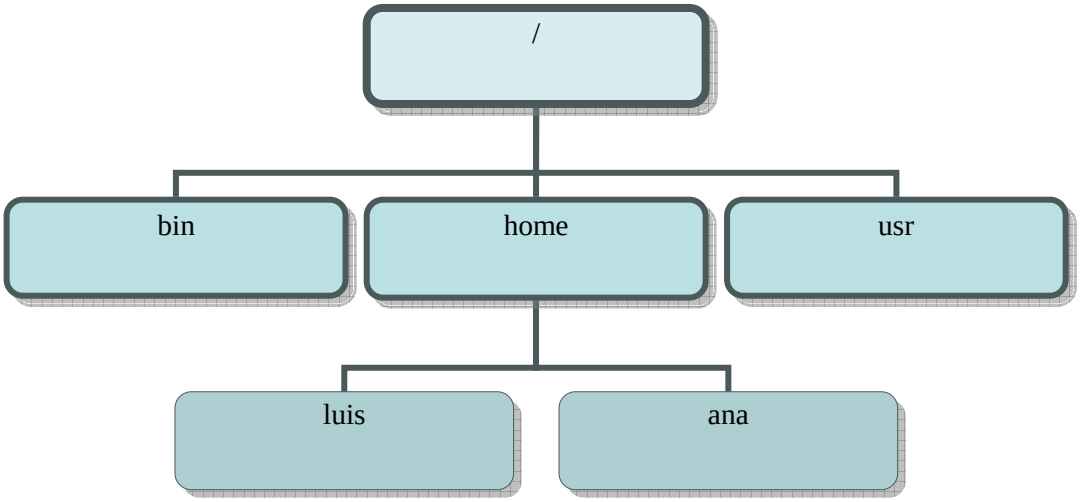
Los archivos en el sistema operativo LINUX se agrupan en directorios jerárquicamente estructurados. Esto significa que cualquier directorio puede tener un directorio padre por encima y uno o varios directorios hijos por debajo. La figura de la página siguiente muestra un ejemplo. Siendo un entorno multiusuario, LINUX asigna a cada uno de sus usuarios un directorio privado desde el que siempre se empieza a trabajar cada vez que iniciamos una nueva sesión. Esos directorios suelen llevar como nombre el del propio usuario y son hijos del directorio `/home`. El propio directorio `/home` es hijo del directorio raíz que está situado en lo alto de la jerarquía de directorios. Este directorio se representa con el símbolo '`/`'.

Orden: ***pwd [opciones]***
Esta orden muestra el nombre del directorio en el que uno se encuentra situado. Se dice que ese directorio es el *directorio de trabajo*.

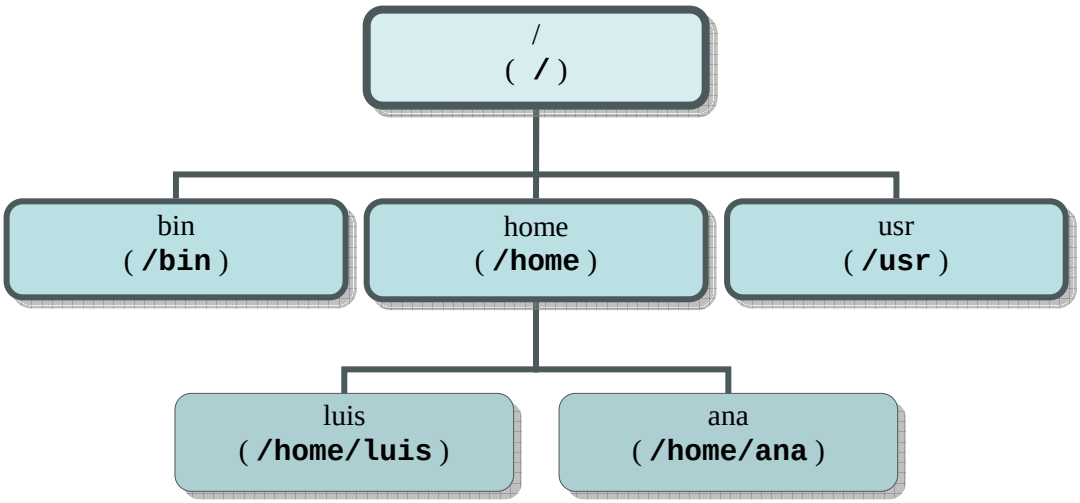
Ejecución: **`pwd`**
Muestra el directorio de trabajo.

Ubicación de los archivos y los directorios: los caminos

Cuando en una orden queremos dar una referencia a un archivo o directorio que se encuentra en el directorio de trabajo, basta con indicar su nombre. Pero si no es el caso, es necesario indicar el lugar en el que se encuentra indicando su camino (en inglés se utiliza la palabra *path*). Diremos que la indicación del camino es absoluta si se toma como punto de referencia al directorio raíz, mientras que la indicación del camino será relativa si se toma como punto de referencia al directorio de trabajo.



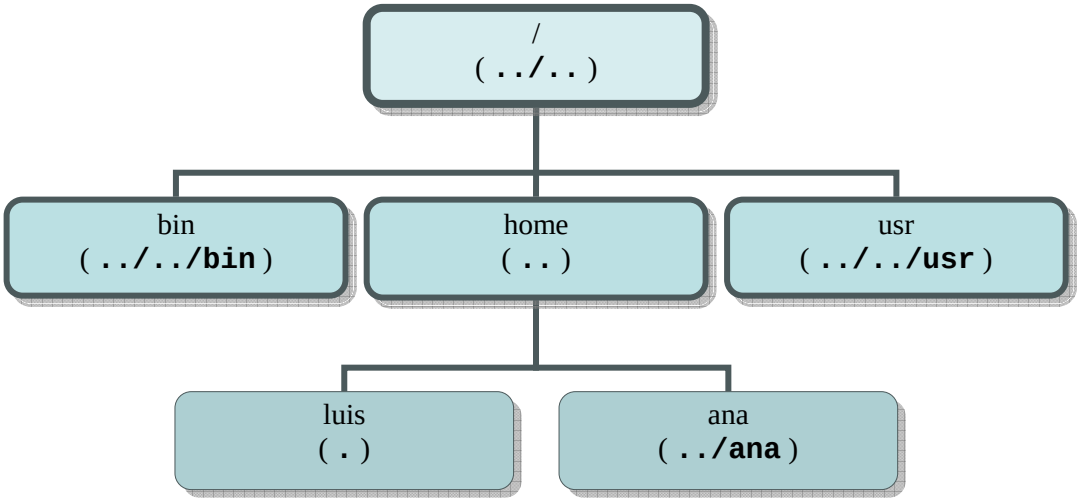
Así pues el camino absoluto se escribe empezando por el directorio raíz '`/`' seguido por los nombres de todos los directorios por los que hay que pasar hasta llegar al archivo o directorio deseado (separándolos todos con el símbolo '/').
La siguiente figura muestra entre paréntesis los caminos absolutos para cada uno de los directorios del ejemplo anterior.



Para el uso de caminos relativos, el símbolo '`.`' representa el directorio de trabajo mientras que '`..`' representa el directorio padre del directorio de trabajo. La siguiente figura (en la página siguiente) nos muestra de nuevo el mismo árbol de directorios pero esta vez especificando entre paréntesis los caminos relativos (en este caso el directorio de trabajo es el directorio `/home/luis`).

Órdenes relativas al sistema de ficheros

Como ya hemos visto anteriormente, en LINUX tenemos una orden que permite visualizar el contenido de los directorios:



*Orden: **ls [opciones] [ficheros]***
Esta orden da un listado del contenido de directorios.

Ejecución: **`ls`**
Muestra el contenido del directorio actual.

Además de la visualización de los contenidos de los directorios es posible cambiar de directorio, crear directorios nuevos o borrar directorios ya existentes.

*Orden: **cd [opciones] [camino]***
Esta orden permite cambiar de directorio especificando un camino.

Ejecución: **`cd`**
En el caso de no poner ningún camino, la orden `cd` cambia el directorio de trabajo al directorio por defecto de cada usuario (evidentemente, el suyo propio).

*Orden: **mkdir [opciones] camino***
Esta orden permite crear directorios nuevos.

Ejecución: **`mkdir temp`**
Crea un subdirectorio denominado `temp` en el directorio de trabajo actual.

*Orden: **rmdir [opciones] camino***
Esta orden permite borrar directorios.

Ejecución: **`rmdir temp`**
Borra el subdirectorio denominado `temp` (esta orden solo funciona en el caso de que el directorio esté vacío).

También es posible copiar, mover, renombrar y borrar archivos. Las órdenes que se utilizan en esos casos son:

Orden: ***cp [opciones] fichero camino***
Esta orden copia los archivos o directorios señalados en el directorio especificado por el camino.

Ejecución: **`cp archivo /tmp`**
Copia el archivo denominado `archivo` al directorio `/tmp`.

Orden: ***mv [opciones] fichero camino***
Esta orden mueve los archivos o directorios señalados al directorio especificado por el camino.

Ejecución: **`mv archivo /tmp`**
Mueve el archivo denominado `archivo` al directorio `/tmp`.

Ejecución: **`mv archivo1 archivo2`**
Esta es otra forma de utilizar la orden `mv`. En este caso la orden se encarga de renombrar el archivo denominado `archivo1` con el nuevo nombre `archivo2`.

Orden: ***rm [opciones] fichero***
Esta orden borra los archivos especificados.

Ejecución: **`rm archivo`**
Borra el archivo denominado `archivo`.

Ejecución: **`rm –R directorio`**
Como hemos visto antes, si un directorio no está vacío no es posible borrarlo con la orden `rmdir`. La orden `rm` nos lo permite con la opción `–R` borrando el contenido del directorio especificado y todos sus subdirectorios de forma recursiva. AVISO: hay que tener mucho cuidado con esta orden.

Ejercicios:
5.- Cambiar el directorio de trabajo por el directorio que tenemos asignados por defecto (utilizando como referencia a nuestro directorio el camino absoluto).
6.- Listar en formato largo (con información adicional para cada archivo y/o directorio más allá de sus nombres) el contenido del directorio de trabajo actual y redirigir *la salida estándar* a un archivo nuevo denominado `listado.txt` .
7.- Crear un nuevo directorio denominado `seguridad`.
8.- Hacer una copia del archivo `listado.txt` al directorio `seguridad`.
9.- Borrar el archivo inexistente `no_existe.txt` (como es de suponer, esta orden dará un error) y redirigir *el error estándar* al archivo `error.txt`.
10.- Renombrar el archivo `error.txt` como `errores.txt`.
11.- Mover el archivo `errores.txt` al directorio `seguridad`.
12.- Borrar el directorio `seguridad`.

Los comodines

Como ya vimos en la página 4 de este guión, es posible utilizar comodines a la hora de especificar nombres de archivos y/o de directorios. Los comodines son unos caracteres especiales que pueden sustituir a nombres y a partes de nombres de los archivos y/o directorios, lo cual facilita la especificación de múltiples nombres como argumentos de una orden:

- '*****': sustituye cualquier secuencia de caracteres
- '**?**': sustituye un único carácter

Ejecución: **`rm a*`**
Borra todos los archivos que empiecen por `a`.

Ejecución: **`ls archivos1?`**
Lista todos los archivos que empiecen por `archivos1` seguido de cualquier carácter.

Permisos

En LINUX, que es un sistema multiusuario, para evitar accesos a archivos y directorios por parte de usuarios no autorizados, se han desarrollado mecanismos de protección. Para cada archivo o directorio, el sistema operativo agrupa todos los usuarios que tienen acceso a la máquina en tres clases básicas:

- Propietario: se refiere al usuario que ha creado el archivo o directorio.
- Grupo: se refiere a todos aquellos usuarios que pertenezcan al mismo grupo de usuarios que el propietario.
- Otros: se refiere a cualquier otro usuario.

Además, para cada clase el sistema operativo asigna tres tipos de permisos de acceso:

- Permiso de lectura: se refiere al permiso para poder visualizar el contenido del archivo o directorio.
- Permiso de escritura: se refiere al permiso para poder modificar el contenido del archivo o directorio.
- Permiso de ejecución: se refiere al permiso para poder ejecutar el archivo o acceder al directorio.

Los permisos de cualquier archivo o directorio se pueden ver ejecutando la orden '`ls –l`', apareciendo en la primera columna con el formato `drwxrwxrwx`. El primer carácter es una '`d`' en el caso de que se esté listando un directorio y '`-`' en el caso de un archivo. A continuación viene los permisos de lectura '`r`', escritura '`w`' y ejecución '`x`' para cada una de las clases siendo los tres primeros para el *propietario*, los tres siguientes para el *grupo* y los tres últimos para los *otros*.

Solo los propietarios pueden cambiar los permisos de un archivo o un directorio. Para poder hacerlo los propietarios deben utilizar la siguiente orden:

Orden: ***chmod [opciones]usuario(s) accion(es) permiso(s) fichero***
Esta orden modifica los permisos de los archivos y/o directorios especificados.

Ejecución: **`chmod ug+rx archivo`**
Añade permisos de lectura y ejecución tanto para el propietario como para los que pertenecen a su grupo en el archivo denominado `archivo`.

Como muestra el ejemplo de ejecución es posible añadir o quitar cualquiera de los permisos a una o varias clases a la vez. Las posibilidades son las siguientes:

- usuarios: propietario (**`u`**), grupo (**`g`**), otros (**`o`**) y/o todos (**`a`**).
- acciones: dar permiso (**`+`**) o quitar permiso (**`-`**)
- permisos: lectura (**`r`**), escritura (**`w`**) y/o ejecución (**`x`**)

Existe otra forma de enumerar las modificaciones que se pretenden efectuar en los permisos de un archivo o directorio. Basta con representar el formato de los permisos `rwx rwx rwx` mediante tres números binarios de tres bits cada uno, de manera que si se quiere conceder un

permiso determinado se le asigna un 1 y si no un 0. De esta forma podemos representar los permisos mediante un número de tres dígitos octales. En ese caso, la sintaxis de la orden `chmod` cambia.

Orden: ***chmod [opciones] permiso_en_octal fichero***
Esta orden modifica los permisos de los archivos y/o directorios especificados.

Ejecución: **`chmod 754 archivo`**
Activa los permisos de lectura, escritura y ejecución para el propietario, de lectura y ejecución para los que pertenezcan al mismo grupo y solo el de lectura para los demás en el archivo denominado `archivo`.

Ejercicios:
13.- Buscar en el directorio `/etc` aquellos archivos que empiecen por "`auto.`" (NOTA: hay un punto al final del nombre).
14.- Ver los permisos de todos esos archivos a la vez y comentar cuales son.
15.- Intentar modificar sus permisos añadiéndoles permisos de escritura para las 3 clases (Comentar cual de las dos formas es la más adecuada). ¿ Porqué da error ? ¿ Qué podríamos hacer para cambiarles los permisos ?