SISTEMAS INFORMÁTICOS UD-3

SISTEMAS OPERATIVOS. GESTION DE ARCHIVOS Y ALMACENAMIENTO

GUÍA DE COMANDOS EN SHELL PARA LA GESTIÓN DE ARCHIVOS

ÍNDICE

[cd (CHANGE DIRECTORY). CAMBIAR DE DIRECTORIO. 3](#_Toc126410014)

[pwd (PRINT WORKING DIRECTORY). VISUALIZACIÓN DEL DIRECTORIO ACTUAL. 3](#_Toc126410015)

[touch Y mkdir (MAKE DIRECTORY). CREACIÓN DE FICHEROS. 3](#_Toc126410016)

[cp (COPY). COPIA DE FICHEROS 3](#_Toc126410017)

[man (MANUAL). MANUAL DE USO 4](#_Toc126410018)

[rm (REMOVE) Y rmdir (REMOVE DIRECTORY). BORRADO DE FICHEROS 5](#_Toc126410019)

[ls (LIST). LISTADO DE FICHEROS 6](#_Toc126410020)

[stat. VISUALIZACIÓN DE INODOS 8](#_Toc126410021)

[ln (LINK). CREACIÓN DE ENLACES DUROS Y BLANDOS 8](#_Toc126410022)

[echo Y cat. IMPRESIÓN DE CONTENIDOS POR CONSOLA 10](#_Toc126410023)

[mv (MOVE). MOVIMIENTO DE FICHEROS 10](#_Toc126410024)

[more Y less. VISUALIZACIÓN DE CONTENIDOS MULTIPÁGINA 11](#_Toc126410025)

[head Y tail. VISUALIZACIÓN DE PARTES DE CONTENIDO 12](#_Toc126410026)

[wc (WORD COUNT). CONTADOR DE PALABRA, LÍNEAS, ETC. 12](#_Toc126410027)

[sort. ORDENAMIENTO DE UN FICHERO 13](#_Toc126410028)

[nano. PROCESADOR DE TEXTO POR CONSOLA 15](#_Toc126410029)

[ENTRADA ESTÁNDAR, DE ERROR Y SALIDA ESTÁNDAR. REDIRECCIONES. 15](#_Toc126410030)

[A. REDIRECCIÓN DE LA SALIDA ESTÁNDAR 15](#_Toc126410031)

[B. REDICCIÓN DE LA ENTRADA ESTÁNDAR 16](#_Toc126410032)

[C. REDIRECCIÓN DE LA SALIDA DE ERROR 17](#_Toc126410033)

[D. REDIRECCIÓN DE LA SALIDA Y EL ERROR ESTÁNDAR AL MISMO DESTINO 17](#_Toc126410034)

[E. REDIRECCIÓN DE LA SALIDA Y EL ERROR ESTÁNDAR DE UNA ORDEN A LA ENTRADA ESTÁNDAR DE OTRA ORDEN 18](#_Toc126410035)

[F. REDIRECCIÓN DE LA SALIDA DE ERROR A LA SALIDA ESTÁNDAR 19](#_Toc126410036)

[cut. DIVIDIR UN FICHERO EN COLUMNAS 19](#_Toc126410037)

[grep. LOCALIZACIÓN DE SECUENCIAS DE CARACTERES 20](#_Toc126410038)

[find. BÚSQUEDA DE ARCHIVOS 22](#_Toc126410039)

# cd (CHANGE DIRECTORY). CAMBIAR DE DIRECTORIO.

El comando cd sirve para cambiar el directorio donde nos encontramos con la terminal. Sigue la siguiente sintaxis: cd [ruta].

Texto

Descripción generada automáticamente

Si simplemente escribimos cd nos llevará a la ruta /home/usuario y siempre que queramos volver al directorio padre del directorio actual nos servirá con escribir cd .. o cd ../ en la terminal.

Imagen que contiene firmar, verde, alimentos, parada

Descripción generada automáticamente

# pwd (PRINT WORKING DIRECTORY). VISUALIZACIÓN DEL DIRECTORIO ACTUAL.

El comando pwd nos sirve para conocer el directorio actual donde nos encontramos:



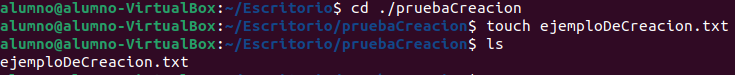
# touch Y mkdir (MAKE DIRECTORY). CREACIÓN DE FICHEROS.

Para la creación de ficheros tenemos 2 opciones:

* El comando mkdir que nos servirá para crear un fichero de tipo directorio y que tiene la siguiente sintaxis: mkdir [archivo].



* El comando touch que nos servirá para crear un fichero de tipo fichero regular (por ejemplo, un documento de texto) y que tiene la siguiente sintaxis: touch [archivo].



# cp (COPY). COPIA DE FICHEROS

El comando cp nos sirve para copiar archivos y carpetas de un directorio a otro. La sintaxis básica es: cp [opciones] lista\_de\_archivos\_origen destino.

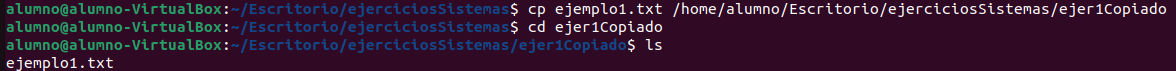
Por ejemplo, vamos a crear un directorio en el Escritorio del sistema, para ello usamos el comando mkdir y dentro de este directorio creamos otro.

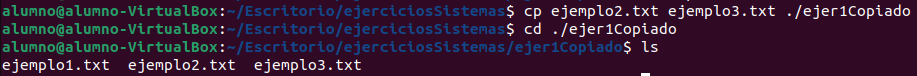


A su vez vamos a crear tres archivos en el primer directorio (ejerciciosSistemas) haciendo uso del comando touch. Se pueden crear archivos 1 a 1 o varios a la vez:



Para copiar archivos también podemos hacerlo 1 a 1 o varios a la vez, de la misma forma podemos hacer referencia a la ruta completa o a una ruta relativa:





Tenemos varios modificadores:

* i 🡪 Solicita información antes de sobrescribir. Por defecto, sobrescribe automáticamente en el destino.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* r o R 🡪 Copia de forma recursiva en directorios. Se pueden copiar directorios también y si el directorio de destino no existe, se crearía automáticamente:

Imagen que contiene Texto

Descripción generada automáticamente

Imagen que contiene Texto

Descripción generada automáticamente

# man (MANUAL). MANUAL DE USO

El comando man nos sirve como manual de utilización de los diferentes comandos de Linux y su sintaxis es: man [comando].

Por ejemplo, si queremos acceder a la información disponible del comando cp debemos introducir el comando man cp.

Texto

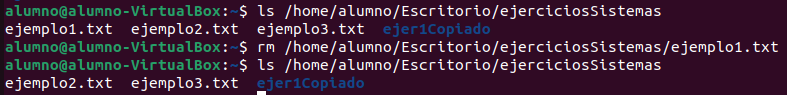
Descripción generada automáticamente

En esta nueva ventana podremos ver información relacionada a la función del comando, a la sintaxis, a las subfunciones, etc.

# rm (REMOVE) Y rmdir (REMOVE DIRECTORY). BORRADO DE FICHEROS

El comando rm cumple la función de eliminar archivos y directorios y tiene la siguiente sintaxis: rm [ruta] [parámetros] [archivo]

Por ejemplo, vamos a eliminar un archivo ejemplo1.txt:

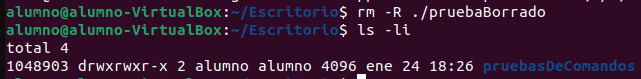


El comando rm tiene una serie de modificadores propios:

* r o R 🡪 Elimina subdirectorios de forma recursiva.

Texto

Descripción generada automáticamente



* i 🡪 Pide confirmación para cada borrado.



* f 🡪 fuerza la eliminación aun estando protegido el archivo contra escritura.

Es un comando que permite emplear los caracteres comodín \* y ? para generar patrones de identificación de ficheros.

* \*: Cualquier cadena de caracteres. Por ejemplo, \*.txt borraría todos los ficheros que tengan la extensión txt.

Texto

Descripción generada automáticamente

* ?: Un único carácter cualquiera, pero debe haberlo, por ejemplo, fichero?.txt borraría fichero1.txt, fichero2.txt, fichero3.txt… pero no fichero.txt; tampoco borraría un fichero1b.txt.

Texto

Descripción generada automáticamente

Al igual que pasa con otros comandos de Linux, se pueden utilizar varios modificadores a la vez, por ejemplo, rm -rf eliminará los subdirectorios de forma recursiva y forzada.

Además, tenemos el comando rmdir que en lugar de borrar ficheros, borra directorios y que sigue una sintaxis similar: rmdir [ruta] [parámetros] [elemento].



# ls (LIST). LISTADO DE FICHEROS

El comando ls sirve para listar los elementos de un sistema de archivos y tiene la siguiente sintaxis: ls [opciones] [ruta]



El comando ls tiene una serie de modificadores propios:

* l 🡪 Muestra un listado en formato largo con la información de permisos, número de enlaces duros, usuario principal, grupo, tamaño y fecha de última modificación.

Imagen de la pantalla de un computador

Descripción generada automáticamente con confianza baja

* d 🡪 Muestra solamente el nombre del subdirectorio sin entrar en él.

Texto

Descripción generada automáticamente

* t 🡪 Ordena por la fecha de modificación.

Imagen de la pantalla de un computador

Descripción generada automáticamente con confianza baja

* a 🡪 Muestra los archivos ocultos.

Interfaz de usuario gráfica, Texto

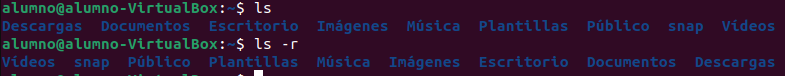
Descripción generada automáticamente

* R 🡪 Hace un listado recursivo entrando en cada uno de los subdirectorios y listándolos.

Texto

Descripción generada automáticamente

* r 🡪 Lista los elementos en orden inverso.



También podemos hacer un ls de dos directorios diferentes a la vez:

Texto

Descripción generada automáticamente

# stat. VISUALIZACIÓN DE INODOS

El comando stat nos sirve para conocer las estadísticas de un archivo o un sistema de archivos. Sigue la siguiente sintaxis: stat [opciones] [archivo].

La información que nos suministra es la relativa al tamaño, el número de bloques que ocupa, la cantidad de enlaces que tiene, el UID, el GID…

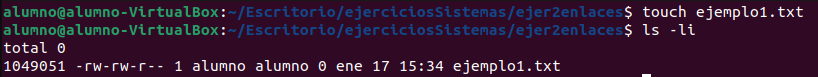
Texto

Descripción generada automáticamente

# ln (LINK). CREACIÓN DE ENLACES DUROS Y BLANDOS

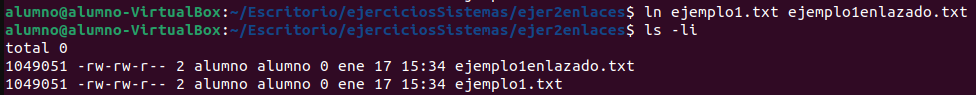
El comando ln nos va a servir para crear enlaces tanto blandos como duros.

Para crear enlaces duros hay que hacer lo siguiente. Primero vamos a crear un nuevo archivo con el comando touch y vamos a ver su número de inodo con el modificador -i del comando ls:

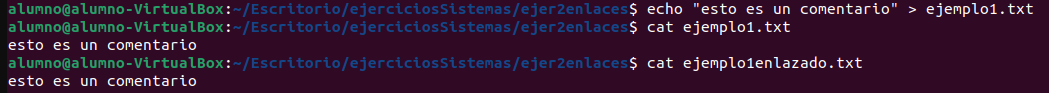


Los primeros dígitos que se ven hacen referencia al inodo del archivo ejemplo1.txt.

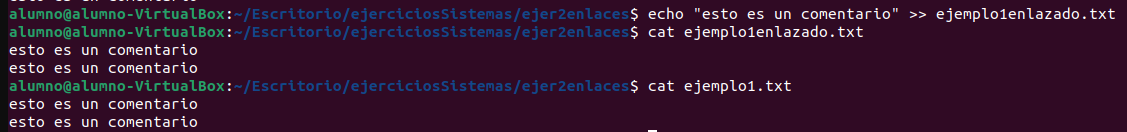
Ahora vamos a crear un enlace duro haciendo uso del comando ln con la siguiente sintaxis: ln [archivo original] [enlace] y si volvemos a hacer un ls -li veremos que comparten el mismo inodo:



Vamos a comprobar ahora el funcionamiento interno escribiendo algo dentro de ambos archivos, por ejemplo, vamos a escribir dentro de ejemplo1.txt:



Y ahora escribiendo dentro de ejemplo1enlazado.txt:



Lo que escribamos en un archivo se va a replicar en el otro archivo porque ambos están enlazados ya que apuntan al mismo contenido de memoria.

En el caso de querer crear enlaces blandos tendremos que usar el modificador -s siguiendo la sintaxis: ln -s [archivoOriginal] [nombreEnlaceBlando].

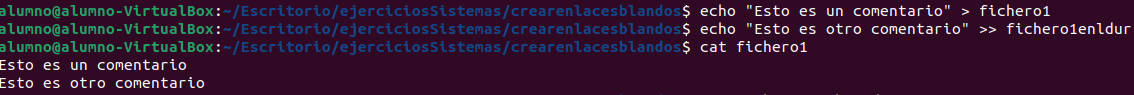
Texto

Descripción generada automáticamente

Como se puede ver enlace blando y fichero original no comparten inodo. Podemos introducir texto en el fichero original y en un enlace duro y comprobar como desde el enlace blando podemos acceder a ese contenido:

Texto

Descripción generada automáticamente





Si borramos el archivo al que apunta el enlace blando, este perderá su utilidad ya no se podrá acceder a la información desde él:

Texto

Descripción generada automáticamente con confianza baja

Aunque la información sigue estando accesible gracias al enlace duro:



# echo Y cat. IMPRESIÓN DE CONTENIDOS POR CONSOLA

El comando echo nos sirve para imprimir mensajes por pantalla o para introducir texto dentro de un archivo siguiendo la sintaxis echo [opción] [string].

Si no escribimos ninguna opción, se imprimirá el mensaje por consola:



El comando cat nos sirve para imprimir por pantalla el texto que se encuentra dentro de un fichero de texto utilizando la sintaxis cat [archivo] [{>|>>|<|<<} archivo]:



La salida puede redireccionar a un archivo mediante:

* > 🡪 Sobrescribe si existe el archivo o, en caso contrario, lo crea.
* >> 🡪 Añade al contenido de un fichero existente o, en caso contrario, lo crea.

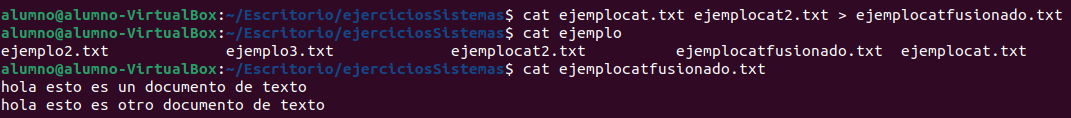
También se puede usar como método de redireccionamiento de archivos, por ejemplo.

* cat > [archivo] 🡪 Genera un archivo nuevo con el nombre introducido que generará como procesador de texto para introducir directamente el texto. Para salir de esta pantalla usar Ctlr + D.

Texto

Descripción generada automáticamente

* cat [archivo] [archivo] > [archivo\_fusionado] 🡪 fusiona los archivos en otro.



# mv (MOVE). MOVIMIENTO DE FICHEROS

Se utiliza el comando **mv** para mover archivos de un lado a otro, es similar al cp pero en este caso en el origen desaparecen los archivos (es parecido a un “cortar”). Se sigue la siguiente sintaxis: mv [opciones] [archivo/s origen] [destino].

Texto

Descripción generada automáticamente

* i 🡪 Solicita confirmación antes de sobrescribir.
* u 🡪 solo mueve archivos o directorios con el mismo nombre entre origen y destino si éstos son más actuales en el origen.

# more Y less. VISUALIZACIÓN DE CONTENIDOS MULTIPÁGINA

Ambos comentarios permiten la visualización de contenidos que ocuparían varias páginas poco a poco.

* more carga completamente el archivo y utiliza la sintaxis more [lista de archivos]:

Texto

Descripción generada automáticamente

* less carga el archivo según se va visualizando y es más rápido. Utiliza la sintaxis less [lista de archivos]:

Texto

Descripción generada automáticamente

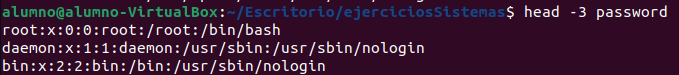
Ambos tienen una serie de controles propios:

* Barra espaciadora 🡪 pasa a la siguiente página.
* Enter 🡪 pasa a la siguiente línea.
* Tecla Q 🡪 sale de la visualización.
* /texto 🡪 busca el texto o expresión regular.
* n 🡪 busca la siguiente coincidencia del texto o expresión regular.
* :n 🡪 pasa al siguiente fichero.
* :p 🡪 retrocede al fichero anterior.

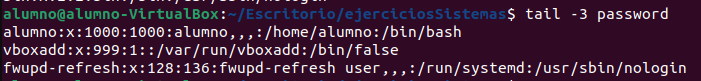
# head Y tail. VISUALIZACIÓN DE PARTES DE CONTENIDO

Sirven para obtener el principio (head) o el final (tail) de un documento. Por defecto muestran las 10 primeras/últimas líneas.

* head utiliza la sintaxis: head [-número líneas] [lista de archivos]



* tail utiliza la sintaxis: tail [-número líneas] [lista de archivos]



Se pueden combinar para conseguir, por ejemplo, la primera línea de las 3 últimas líneas de un documento:



# wc (WORD COUNT). CONTADOR DE PALABRA, LÍNEAS, ETC.

El comando wc nos sirve para conseguir información sobre el número de líneas, el número de palabras, etc. de un documento.

Sigue la sintaxis wc [opciones] [lista de archivos], por defecto, muestra lo siguiente:



El primer valor es el número de líneas, el 2º el número de palabras, el tercero el tamaño en bytes y el último el nombre del archivo.

El comando tiene una serie de opciones:

* l 🡪 muestra únicamente el número de líneas.



* w 🡪 muestra únicamente el número de palabras.



* c 🡪 muestra únicamente el número de bytes.



* L 🡪 muestra la longitud de la línea más larga.



# sort. ORDENAMIENTO DE UN FICHERO

El comando sort sirve para ordenar el texto de un documento.

Su sintaxis es la siguiente: sort [opciones] [lista de archivos].

Si no se realiza ninguna opción, ordena siguiendo el orden establecido por los caracteres ASCII, donde las letras minúsculas tienen un orden mayor que las mayúsculas.

Texto

Descripción generada automáticamente

Tiene una serie de opciones:

* f 🡪 Ignora mayúsculas y minúsculas.
* r 🡪 Invierte el orden.

Texto

Descripción generada automáticamente

* n 🡪 ordena numéricamente en lugar de alfabéticamente.

Texto

Descripción generada automáticamente

Sin la opción toma los números como secuencia de caracteres y no como números.

Texto

Descripción generada automáticamente

* u 🡪 elimina entradas repetidas.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Una palabra idéntica, pero con diferente combinación de mayúsculas y minúsculas se considera una palabra diferente.

Además de estas opciones el comando sort tiene 2 modificadores especiales:

* t 🡪 indica un delimitador.
* k 🡪 indica el número de campo por el que se va a realizar la ordenación. Por defecto, toma los espacios, tabuladores y carácter de fin de línea como delimitadores de campo.

Si lo queremos usar debemos seguir la siguiente sintaxis: sort [opciones] [-t <tipo de delimitador>] [-k <número del campo>] lista de archivos.

Por ejemplo, si queremos ordenar los usuarios del sistema utilizando el archivo password, ordenándolos por su número tendremos que hacer lo siguiente:

Texto

Descripción generada automáticamente

El delimitador que separa las palabras son los dos puntos (“:”) y la columna que nos interesa es la tercera.

# nano. PROCESADOR DE TEXTO POR CONSOLA

El comando nano despliega un procesador de textos dentro de la terminal que nos permite crear archivos de texto.

Texto

Descripción generada automáticamente

# ENTRADA ESTÁNDAR, DE ERROR Y SALIDA ESTÁNDAR. REDIRECCIONES.

El SO asigna automáticamente a cada comando entradas y salidas estándar asociadas a los flujos de entrada y salida, respectivamente. Por defecto se asignan tres ficheros:

* Entrada estándar 🡪 stdin.
* Salida estándar 🡪 stdout.
* Salida de errores 🡪 stderr.

Estos flujos se identifican por un número o descriptor de fichero: 0, 1 y 2.

Normalmente la entrada estándar está asociada con el teclado y la salida estándar y la salida de errores estándar con la pantalla. Muchos comandos pueden tomar la entrada o salida estándar de otros ficheros.

El comportamiento normal del flujo de un comando se puede “alterar” mediante operadores que redireccionan su flujo a otros comandos, dispositivos o ficheros.

## REDIRECCIÓN DE LA SALIDA ESTÁNDAR

Se utilizan el operador “>” para volcar la salida de una orden sobre un fichero en lugar de a la salida estándar siguiendo la sintaxis [orden] > [fichero].

Texto

Descripción generada automáticamente

En caso de no existir el fichero, se crea uno nuevo, y si existe, se sobrescribe su contenido. También se pude utilizar “1>” ya que “1” hace referencia a la salida estándar.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Con “>>” se añade el contenido al final del fichero sin sobrescribirlo siguiendo la sintaxis [orden] >> [fichero].



Texto

Descripción generada automáticamente

## REDICCIÓN DE LA ENTRADA ESTÁNDAR

Se utiliza el operador “<” para redireccionar un fichero como entrada de una orden en lugar de la entrada estándar siguiendo la sintaxis [orden] < [fichero].

Texto

Descripción generada automáticamente

Existe una redirección particular que permite introducir texto hasta que se encuentre una línea únicamente con el delimitador establecido utilizando la sintaxis [orden] << [delimitador].

Texto

Descripción generada automáticamente

## REDIRECCIÓN DE LA SALIDA DE ERROR

Se utiliza el operador “2>” para redireccionar la salida de error estándar a un fichero siguiendo la sintaxis [orden] < [fichero].

También se puede utilizar el operador “2>” para añadir contenido al fichero.´

Por ejemplo, vamos a solicitar el listado de un directorio inexistente:

Texto

Descripción generada automáticamente con confianza baja

El comando devuelve por pantalla (salida estándar) el error, pero podemos redireccionarlo:



Tanto salida estándar como de errores se pueden combinar en un único comando para diferenciar salidas normales y erróneas, por ejemplo, si intentamos hacer un listado recursivo (ls -R) del directorio raíz (/) vamos a recibir varios errores de permiso de acceso, si hacemos un redireccionamiento estándar a un fichero listado.txt y un redireccionamiento de errores a un fichero errores.txt, separaremos los accesos correctos (listado) de los denegados (errores).

Texto

Descripción generada automáticamente

## REDIRECCIÓN DE LA SALIDA Y EL ERROR ESTÁNDAR AL MISMO DESTINO

Se utiliza el operador “&>” para redireccionar ambas salidas a un mismo destino con la sintaxis [orden] &> [fichero] u [orden] &>> [fichero].

Por ejemplo, en un caso como el anterior, si queremos hacer un listado recursivo del directorio raíz e incluir errores y accesos correctos en un mismo archivo haríamos lo siguiente:



## REDIRECCIÓN DE LA SALIDA Y EL ERROR ESTÁNDAR DE UNA ORDEN A LA ENTRADA ESTÁNDAR DE OTRA ORDEN

Se utiliza el operador “tubería” o “pipe” (“|”) para concatenar salidas estándar con entradas estándar entre órdenes. Para ello se sigue la sintaxis [orden A] | [orden B].



Con este comando lo que hacemos es un listado del directorio /bin (orden A) y calculamos las líneas con el comando wc (orden B), es decir, hemos mandado la salida de la orden A como entrada de la orden B que terminada dando una salida a pantalla.

Con el operador “|&” podemos redireccionar tanto la salida estándar como la de error, se sigue la sintaxis [orden A] |& [orden B].

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

El comando lo que hace es en primer lugar hacer un listado recursivo del directorio raíz y mandar por la salida estándar todas las peticiones aceptadas (no de error) a wc -l para conocer el número de líneas. Las salidas de error se muestran por pantalla.



Ahora el comando redirecciona tanto las salidas estándar como las salidas de error.

Si se quiere además enviar la salida a un fichero usamos el operador “tee” siguiendo esta sintaxis [orden A] | [tee fichero] | [orden B].





## REDIRECCIÓN DE LA SALIDA DE ERROR A LA SALIDA ESTÁNDAR

Se utiliza el operador “2>&1” para concatenar salidas estándar con salidas de error entre órdenes.



# cut. DIVIDIR UN FICHERO EN COLUMNAS

El comando “cut” nos va a servir para dividir un fichero o cadena de caracteres en columnas.

Se utiliza la siguiente sintaxis: cut -c <lista\_caracteres> | -f <lista\_columnas> [-d <delimitador>] fichero.

Es un comando que cuenta con 2 opciones:

* c 🡪 Corta por caracteres especificados en el comando.



* f 🡪 Corta por campos establecidos en el comando. Por defecto los delimitadores son: espacio, tabulador o espacio fin de línea.

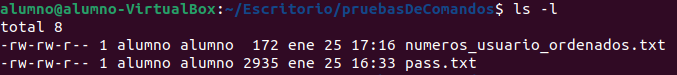
Texto

Descripción generada automáticamente

Con este comando sacamos los números de identificación de todos los usuarios del sistema. Esta orden, como cualquier otra, se puede concatenar con otra orden, por ejemplo:

Texto

Descripción generada automáticamente



No se puede ejecutar -c y -f de forma conjunta.

# grep. LOCALIZACIÓN DE SECUENCIAS DE CARACTERES

El comando grep no sirve para buscar un patrón de caracteres en uno o varios ficheros, mostrando las líneas o ficheros (depende de la opción seleccionada) donde se encuentren coincidencias.

Se sigue la sintaxis: grep [opción] patrón fichero1 [fichero2…].

* l 🡪 sólo muestra los ficheros que contienen el patrón especificado.

Texto

Descripción generada automáticamente

* i 🡪 elimina la distinción entre mayúsculas y minúsculas.

Texto

Descripción generada automáticamente

En el primer caso -i mostraría las ocurrencias de Madrid o madrid y en el caso de la combinación -li nos informaría de los archivos donde se han encontrado las ocurrencias.

* c 🡪 muestra el número de líneas totales que cumplen con el patrón para cada fichero.



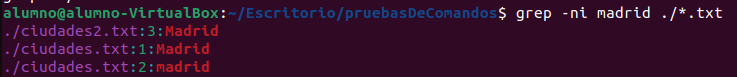


* w 🡪 localiza el patrón como palabra y no como parte de una cadena de texto.

Texto

Descripción generada automáticamente

* n 🡪 imprime el número de línea del patrón localizado.



* v 🡪 busca líneas que no contengan el patrón especificado.

Texto

Descripción generada automáticamente

Al igual que el resto de las opciones se puede combinar con otras opciones para dar lugar a otras respuestas:

Texto

Descripción generada automáticamente

En combinación estas opciones dan lugar a un comando muy potente, por ejemplo, si queremos buscar todas las ocurrencias de mayúsculas y minúsculas de la palabra “Daemon” en el fichero passwd del directorio /etc, haremos lo siguiente:

Texto

Descripción generada automáticamente

Si queremos imprimir el número de líneas donde aparece combinaremos lo anterior con la opción c:



Si queremos imprimir el número de líneas donde no aparece la combinación “daemon:” haremos lo siguiente:



Además, grep nos permite realizar búsquedas con las llamadas **expresiones regulares** que tienen dos formas comunes: **básica** y **extendida.**

|  |  |
| --- | --- |
| **CARACTERES REGEX BÁSICOS** | |
| . | Cualquier carácter individual. |
| [] | Cualquier carácter especificado. |
| [^] | No el carácter especificado. |
| \* | Cero o más de los caracteres previos. |
| ^ | Si ^ es el primer carácter en el patrón, entonces el patrón debe estar al principio de la línea que debe coincidir, de lo contrario sólo un literal ^. |
| $ | Si $ es el último carácter en el patrón, entonces el patrón debe estar al final de la línea que debe coincidir, de lo contrario sólo un literal $. |
| **CARACTERES REGEX AVANZADOS** | |
| + | Uno o más de los patrones previos. |
| ? | El patrón anterior es opcional. |
| {} | Especificar las coincidencias mínimas, máximas o exactas del patrón anterior. |
| | | Alternancia (es una o lógica) |
| () | Permite crear grupos. |

Para el uso de los avanzados es necesario utilizar la opción -E en el grep.

# find. BÚSQUEDA DE ARCHIVOS

Para realizar búsquedas de archivos en Linux se utiliza el comando find con la sintaxis: find [ruta] [criterio de búsqueda] [acción].

Se puede indicar varias rutas o ninguna (en este caso la búsqueda se realizaría sobre el directorio actual en el que se encuentre el usuario).

El resultado y los errores se envían a las salidas estándares por defecto. En caso de no indicar criterio no se hará ningún tipo de filtro.

Existen diferentes criterios:

* **Por nombre del fichero.** Se utiliza la opción name (o iname si no queremos distinguir entre mayúsculas y minúsculas).

Por ejemplo:

find . -name “t\*.txt”

Servirá para buscar todos los archivos que tengan un nombre que comience por t y la extensión .txt.

find . -iname “test\*.txt”

Servirá para buscar todos los archivos que tengan un nombre (en mayúsculas o minúsculas) que comience por test y tenga la extensión .txt.

Texto

Descripción generada automáticamente

* **Por nivel de profundidad en subdirectorios.** Se utilizan las opciones
  + **maxdepth <número>** para determinar hasta qué nivel de profundidad se quiere buscar.
  + **mindepth <número>** para determinar a partir de qué nivel de profundidad se quiere buscar.

Por defecto busca a partir del directorio actual y de forma recursiva.

Por ejemplo,

find . -maxdepth 1 -name texto.txt

Buscará en el directorio actual los archivos con el nombre texto.txt.

find . -mindepth 2 -name texto.txt

Buscará a partir de los subdirectorios del directorio desde el que se hace la llamada.

find . -maxdepth 2 -mindepth 2 -name texto.txt

Buscará únicamente en los subdirectorios del directorio desde el que se hace la llamada.

* **Por tiempos de acceso, modificación y cambio.** Se utilizan las opciones:
  + **m** para la última modificación.
  + **a** para el último acceso.

En ambos casos hay que especificar si son minutos (min) o días (time).

Por ejemplo,

find . -amin -2

Buscará los archivos que hayan sido accedidos hace menos de 2 minutos.

find . -mtime -1

Buscará los archivos que hayan sido modificados hace menos de 1 día.



* **Comparación de ficheros.** Busca ficheros que se modificaron o se accedieron más recientemente que otro de referencia. Tiene dos opciones:
  + **anewer <fichero referencia>** para ficheros cuya fecha de acceso es más reciente.
  + **newer <fichero referencia>** para ficheros que se modificaron más recientemente.

Por ejemplo,

find . -anewer notas.txt

Buscará los archivos que se hayan accedido después que notas.txt.

find . -newer notas.txt

Buscará los archivos que se modificaron más recientemente que notas.txt.



* **Por tamaños.** Se utiliza la opción size seguido de la opción de tamaño (k: kilobytes, M: megabytes, G: gigabytes).

Por ejemplo:

find . -size +1G

Buscará archivos que sean de un peso mayor a 1Gb.

find . -size -1G

Buscará archivos que sean de un peso inferior a 1Gb.

find . -size 1G

Buscará archivos que sean de exactamente 1Gb.

Texto

Descripción generada automáticamente

* **Por tipo de fichero.** Se utiliza la opción type seguido del tipo de enlace a buscar (l: enlace simbólico, d: directorio, f: fichero regular).

Por ejemplo:

find . -type d

Buscará en el directorio actual todos los archivos que sean de tipo directorio.

* **Por permisos.** Se utiliza la opción perm seguido de:
  + -: Debe contener al menos los permisos indicados.
  + /: Debe tener alguno de los permisos indicados.

Si no se indica nada, los permisos deben ser idénticos a los indicados.

find ./dir1 -perm 644

Buscará en el directorio dir1 los archivos que tengan los siguientes permisos:

1. Permisos de propietario: Lectoescritura.
2. Permisos del grupo principal del propietario: Lectura.
3. Permisos del resto de usuarios: Lectura.



* **Otras opciones:**
  + Por **usuario**. Se usa la opción user.
  + Por **inodo**. Se usa la opción inum.
  + Por **UID**. Se usa la opción uid.



* + Por **GID**. Se usa la opción gid.
  + Por estar **vacío**. Se usa la opción empty.

Se pueden combinar opciones de búsquedas con “and”, “or” y “not”.

Texto

Descripción generada automáticamente

Por ejemplo:

find ./dir1 -perm /644 -a -type d

Buscará en dir1 los archivos que tengan alguno de los permisos siguientes:

1. Permisos de propietario: Lectoescritura.
2. Permisos del grupo principal del propietario: Lectura.
3. Permisos del resto de usuarios: Lectura.

Y además sean de tipo directorio.

find ./dir1 -perm /644 -o -perm 755

Buscará en dir1 los archivos que tengan alguno de los permisos siguientes:

1. Permisos de propietario: Lectoescritura.
2. Permisos del grupo principal del propietario: Lectura.
3. Permisos del resto de usuarios: Lectura.

O que tengan exactamente los permisos siguientes:

1. Permisos de propietario: Lectoescritura y ejecución.
2. Permisos del grupo principal del propietario: Lectura y ejecución.
3. Permisos del resto de usuarios: Lectura y ejecución.