MANIPULACIÓN DE ARCHIVOS DE TEXTO.

Ingrese a un shell de línea de comandos de su equipo .

# Determinando el tipo de un archivo

En muchas ocasiones es extremadamente útil determinar que clase de contenido tiene un archivo antes de utilizarlo. Para determinar el tipo de un archivo determinado se utiliza el comando **file**.

Sintaxis: **file *<opciones> <archivo>***

* Ingrese el comando **file --help** y a continuación <ENTER>. **(1)**¿Para qué sirven las opciones del comando file?

-v, --version información de la versión de salida y salir

-m, --magic-file LIST usa LIST como una lista separada por dos puntos de archivos de números mágicos   
-z, --uncompress intenta mirar dentro de los archivos comprimidos

-Z, --uncompress-noreport solo imprime el contenido de los archivos comprimidos

-b, --brief no anteponga nombres de archivo a las líneas de salida

-c, --checking-printout imprimir la forma analizada del archivo mágico, usar en

conjunto con -m para depurar un nuevo archivo mágico

-e, --exclude TEST excluir PRUEBA de la lista de pruebas a ser

realizado para el archivo. Las pruebas válidas son: apptype, ascii, cdf, compress, csv, elf, encoding, soft, tar, json, text, tokens

-f, --files-from FILE leer los nombres de archivo a examinar desde FILE

-F, --separator STRING usa una cadena como separador en lugar de `:'

-i, --mime salida de cadenas de tipo MIME (--mime-type y

--mime-encoding)

--apple salida de Apple CREATOR/TYPE

--extension generar una lista de extensiones separadas por barras

--mime-type generar el tipo MIME

--mime-encoding generar la codificación MIME

-k, --keep-going no te detengas en el primer partido

-l, --list enumerar la fuerza mágica

-L, --dereference Seguir enlaces simbólicos (predeterminado si se establece POSIXLY\_CORRECT)

-h, --no-dereference no seguir enlaces simbólicos (predeterminado si POSIXLY\_CORRECT no está setteado) (predeterminado)

-n, --no-buffer no almacenar en búfer la salida

-N, --no-pad no rellenar la salida

-0, --print0 terminar los nombres de archivo con ASCII NUL

-p, --preserve-date conservar los tiempos de acceso en lo archivos

-P, --parameter establecer límites de parámetros del motor de archivos

indir 15 límite de recurrencia para direccionamiento indirecto

name Límite de 30 usos para nombre/uso de magia

elf\_notes 256 notas máximas de ELF procesadas

elf\_phnum 128 secciones max ELF prog procesadas

elf\_shnum 32768 max ELF sections processed

-r, --raw 32768 secciones max ELF procesadas

-s, --special-files trata los archivos especiales (dispositivos de bloque/char) como archivos ordinaries

-S, --no-sandbox deshabilitar sandboxing de llamadas al sistema

-C, --compile el archivo de compilación especificado por -m

-d, --debug imprime mensajes de depuración

- Ingrese el comando **file /etc/passwd** y a continuación <ENTER>. **(2)**¿De qué tipo de archivo se trata?

Se trata de un archivo de tipo “ASCII TEXT”

* Ingrese el comando **file /bin/bash** y a continuación <ENTER>. **(3)**¿Y este archivo?

* Ingrese el comando **file /dev/tty1** y a continuación <ENTER>. **(4)**¿Qué tipo de archivo es?   
  Es un archivo de tipo ELF
* **(5)**¿De qué tipo es el archivo **/dev/sda**? ¿Y el archivo **/home**? ¿Y el archivo /**dev/psaux**?

El archivo sda es de tipo “bloque especial”, el home es de tipo directorio, y el archivo psaux es de tipo “carácter especial”.

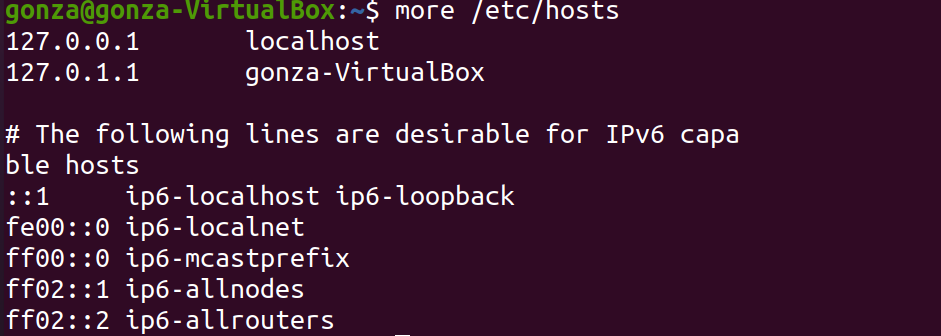
# Mostrando por pantalla el contenido de un archivo de texto

En incontables ocasiones el usuario se verá en la necesidad de ver por pantalla de manera rápida y simple el contenido de un archivo de texto dado.

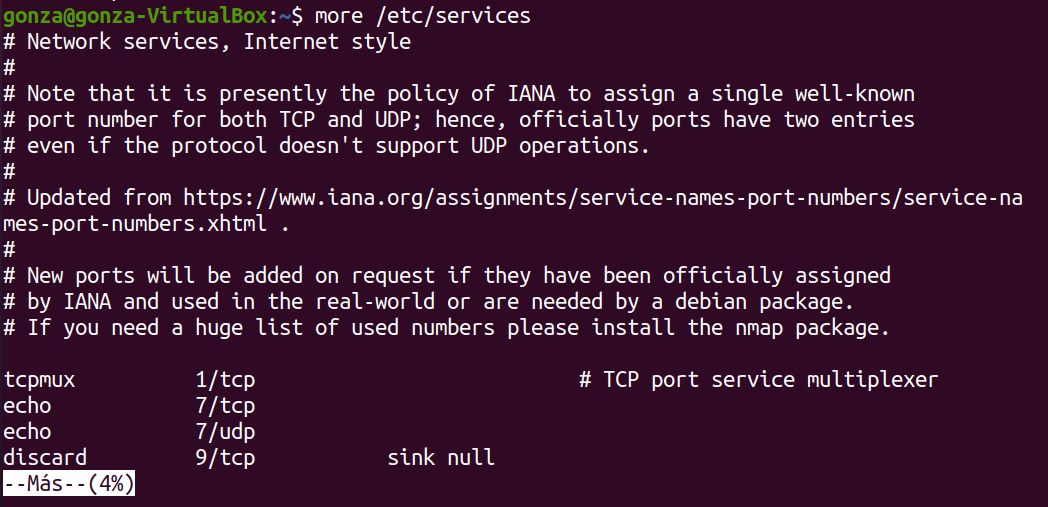
Una de las posibles alternativas es el uso del comando **more**. El comando more tiene la finalidad de mostrar el contenido de un archivo de texto por pantalla de manera paginada (deteniéndose cada vez que se llena la pantalla).

Sintaxis: **more *<opciones> <archivo>***

* Ingrese el comando **more /etc/hosts** y a continuación <ENTER>. **(6)**¿Ve el contenido del archivo?



* Ahora Ingrese el comando **more /etc/services** y a continuación <ENTER>. **(7)** ¿Ve el contenido del archivo? ¿En qué se diferencia el comportamiento del comando more? - Presione la barra espaciadora. Compárelo con el efecto de presionar <ENTER>.

****

La diferencia es que no se mostro todo el contenido ya que se lleno la pantalla, solo mostro el 4% del archivo, al presionar enter, muestra una línea mas del texto, si se sigue presionando enter, podes seguir avanzando.

Otra manera de ver el contenido de un archivo de texto es el uso del comando **less**. El comando less tiene la finalidad de mostrar el contenido de un archivo de texto por pantalla de manera paginada (deteniéndose cada vez que se llena la pantalla). El comando less permite ver la pantalla siguiente con las mismas teclas que more y, si lo deseamos, a diferencia de more, podemos ver la pantalla anterior usando las teclas del cursor o bien mediante **<b>** e **<y>**.

* Ahora Ingrese el comando **less /etc/services** y a continuación <ENTER>. **(8)**¿En qué se diferencia del comportamiento del comando more?

A diferencia del comando more, si bien muestra el contenido, como diferencia principal, no muestra el porcentaje del texto mostrado en la pantalla, la otra diferencia es que si se preciona la tecla con la flecha hacia abajo, podes seguir bajando en el texto, también podes hacerlo con la rueda del more, lo mismo si queres retroceder, podes subir con la flecha hacia arriba o directamente con la rueda del mouse.

# Mostrando por pantalla sólo parte del contenido de un archivo de texto

Cuando trabajemos con archivos de texto cuyo tamaño sea considerable (esto es algo realmente común en las tareas habituales de un administrador de servidores Linux) en ocasiones sólo deseamos ver parte de un archivo de texto (usualmente las *n* primeras líneas o las **n** últimas líneas).

Visualizando el comienzo de un archivo

Para visualizar las n primeras líneas de un archivo de texto se utiliza el comando **head**.

Sintaxis: **head *<opciones> <archivo>***

* Ingrese el comando **head --help** y a continuación <ENTER>. ¿Para qué sirven las opciones del comando head?

-c, --bytes=[-]NUM imprime los primeros NUM bytes de cada archivo;

con el '-' inicial, imprima todo menos el último

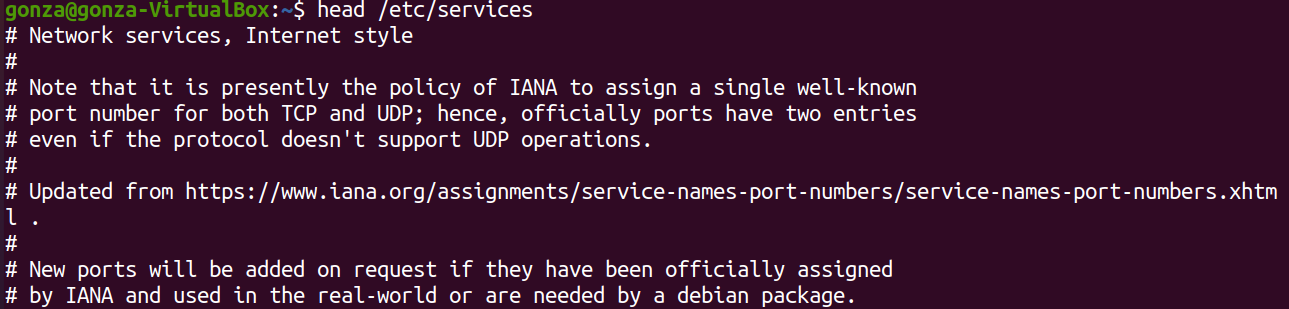
NUM bytes de cada archivo

-n, --lines=[-]NUM imprime las primeras NUM líneas en lugar de las primeras 10;

con el '-' inicial, imprima todo menos el último

NUM líneas de cada archivo

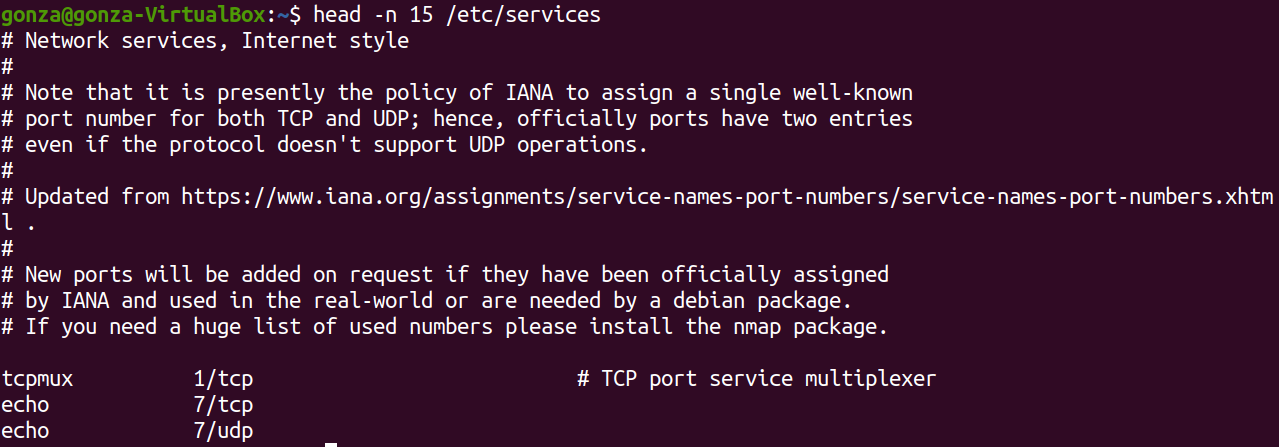
* Ingrese el comando **head /etc/services** y a continuación <ENTER>. **(9)**¿Cuántas líneas del archivo /etc/services le mostró?



El comando head, mostro 11 lineas del archivo services

* Ahora Ingrese el comando **head -n 15 /etc/services** y a continuación <ENTER>.

**(10)**¿Cuántas líneas del archivo /etc/services le mostró ahora?



Mostro la cantidad de líneas que se indicaron, las cuales fueron 15

# Visualizando el final de un archivo

Para visualizar las n últimas líneas de un archivo de texto se utiliza el comando **tail**.

Sintaxis: **tail *<opciones> <archivo>***

* Ingrese el comando **tail --help** y a continuación <ENTER>. ¿Para qué sirven las opciones del comando tail?

-c, --bytes=[+]NUM generar los últimos NUM bytes; o use -c +NUM para

salida que comienza con el byte NUM de cada archivo

-f, --follow[={name|descriptor}] muestra a medida que el fichero crece;

sin opción significa 'descriptor'

-F lo mismo que --follow=name --retry

-n, --lines=[+]NUM genera las últimas NUM líneas, en lugar de las últimas 10;

o use -n +NUM para generar una salida que comience con la línea NUM

--max-unchanged-stats=N   
 con --follow=nombre, vuelve a abrir un ARCHIVO que no tiene

cambio de tamaño después de N (predeterminado 5) iteraciones

para ver si ha sido desvinculado o renombrado

(este es el caso habitual de los archivos de registro rotados);

con inotify, esta opción rara vez es útil --pid=PID con -f, termina después de que el ID del proceso, PID, muere

-q, --quiet, --silent no presenta cabeceras para cada fichero

--retry sigue intentando abrir el fichero si es inaccesible

-s, --sleep-interval=N con -f, espera aproximadamente N segundos entre

iteraciones (por omisión 1.0);

con inotify y --pid=P, comprueba el proceso P al

menos una vez cada N segundos

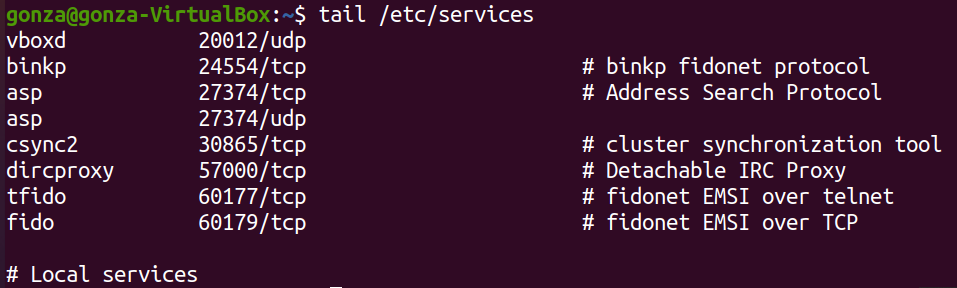
-v, --verbose presenta siempre las cabeceras para cada fichero

-z, --zero-terminated line delimiter is NUL, not newline

--help muestra esta ayuda y finaliza

--version informa de la versión y finaliza

* Ingrese el comando **tail /etc/services** y a continuación <ENTER>. **(11)**¿Cuántas líneas del archivo **/etc/services** le mostró?

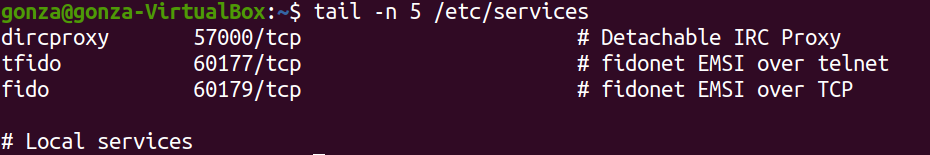


Mostro un total de 11 lineas, contando el salto de línea antes de la última línea

* **(12)** ¿Cómo haría para que el comando tail muestre las últimas **n** líneas?

Si se quisiera que tail muestre N líneas la sintaxis seria: tail -n [CANTIDAD DE LINEAS] [ARCHIVO]

Ejemplo:

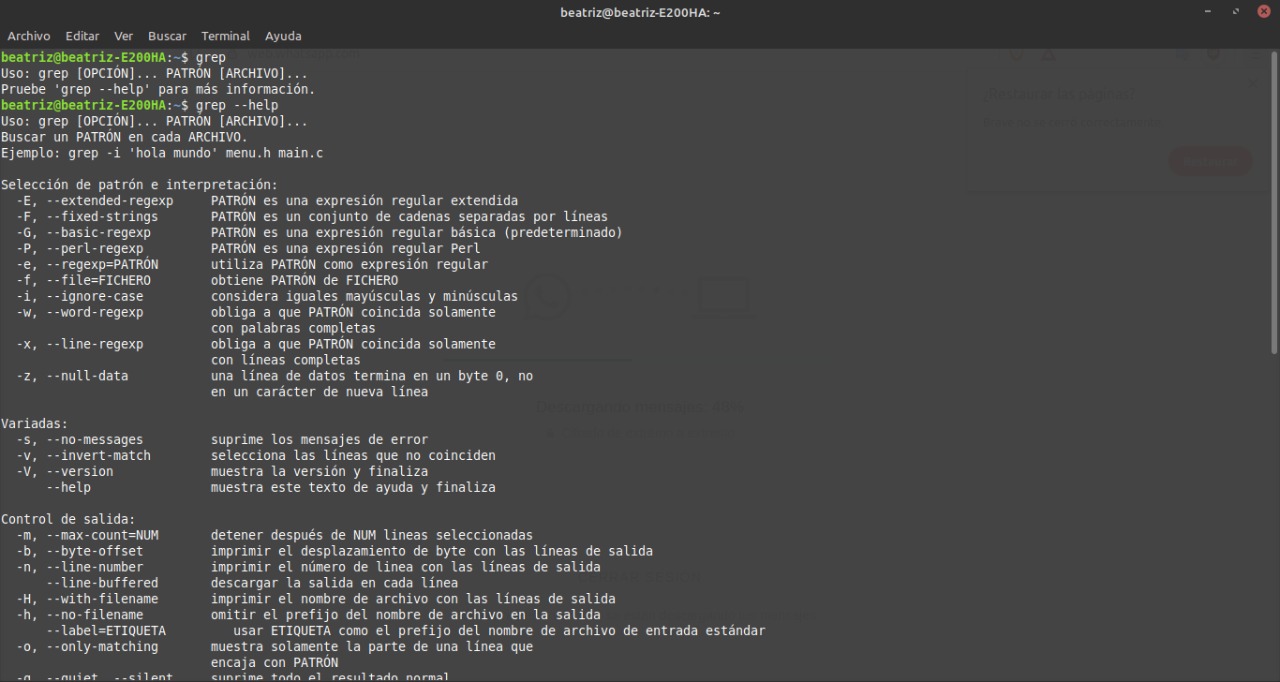


# Buscando patrones dentro de un archivo de texto

Muchas veces necesitamos encontrar las líneas de un archivo de texto que contienen una determinada subcadena o patrón de texto. Para mostrar las líneas que contienen un patrón de texto se utiliza el comando **grep**.

Sintaxis: **grep *<opciones> <patrón> <archivo>***

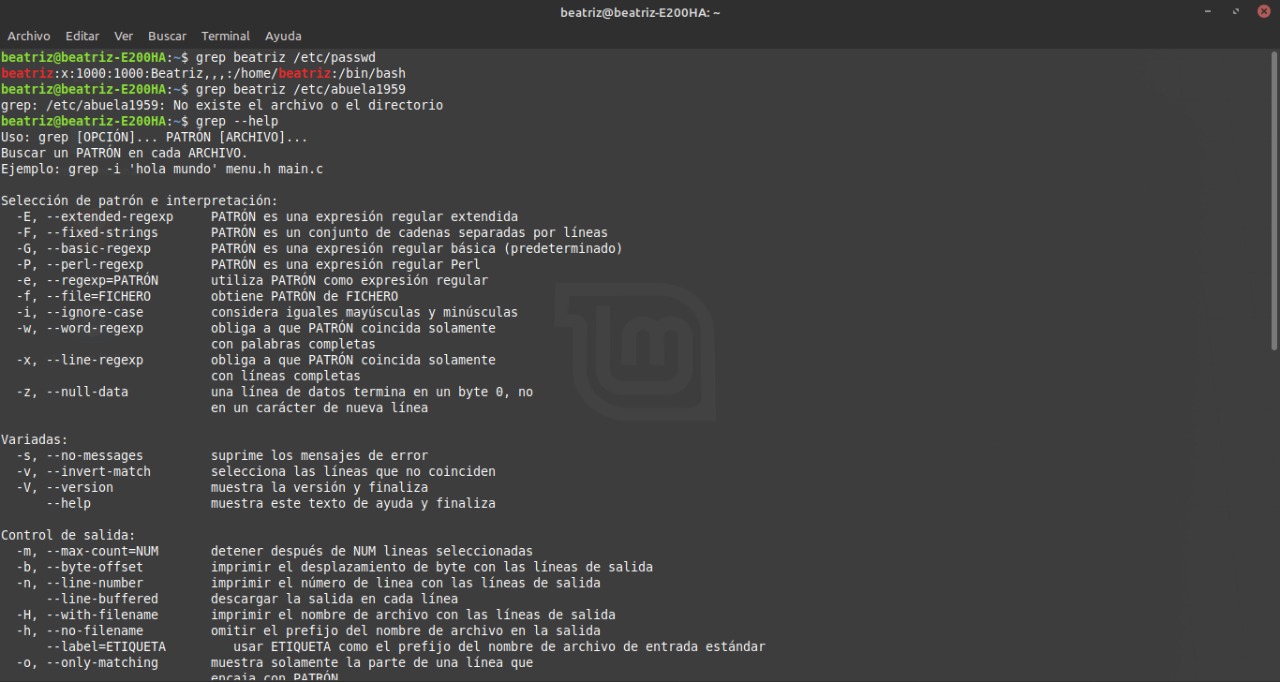
* Ingrese el comando **grep --help** y a continuación <ENTER>. ¿Para qué sirven las opciones del comando grep?



* Ingrese el comando **grep <nombre\_usuario> /etc/passwd** y a continuación <ENTER>. **(13)**¿Qué haría este comando?¿Qué información mostró?¿Porqué mostró solamente la línea alumno?

Este comando muestra los datos del usuario que designamos en la sentencia del comando.

Muestra solamente la línea del usuario ya que esta especificado en el comando.



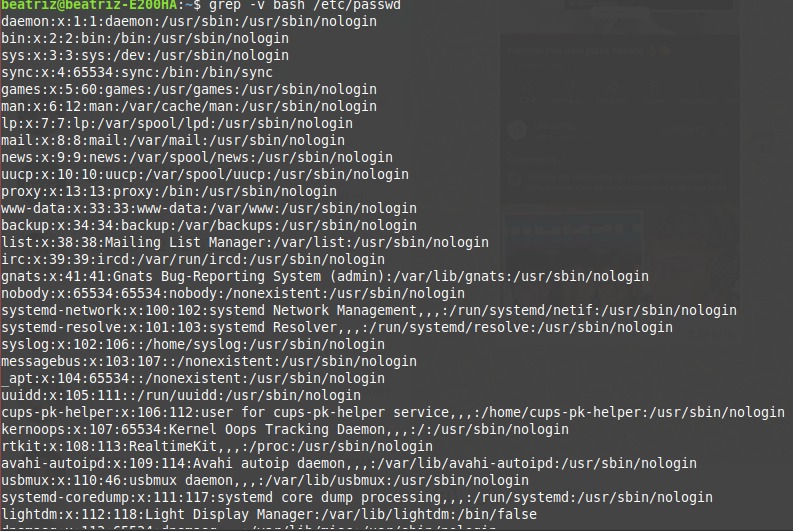
* Ingrese el comando **grep bash /etc/passwd** y a continuación <ENTER>. **(14)**¿Qué haría este comando?¿Qué información mostró?

Muestra todos los directorios que tengan el patron bash en comun.



* Ahora el comando **grep –v bash /etc/passwd** y a continuación <ENTER>. **(15)**¿Qué haría este comando?¿Qué información mostró?¿Nota la diferencia?

Este comando muestra las demás líneas que no coinciden con el comando bash.



# Contando las líneas/palabras/bytes de un archivo

Para contar cuántas líneas/palabras/bytes tiene un archivo de texto se utiliza el comando **wc**.

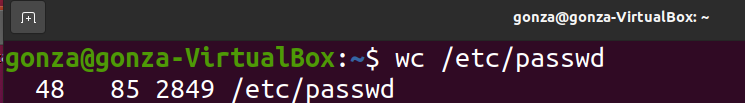
Sintaxis: ***wc <opciones> <archivo>***

* Ingrese el comando **wc --help** y a continuación <ENTER>. ¿Para qué sirven las opciones del comando wc?

permite realizar diferentes conteos desde la entrada estándar, ya sea de palabras, caracteres o saltos de líneas.

* Ingrese el comando **wc /etc/passwd** y a continuación <ENTER>. **(16)**¿Qué haría este comando? ¿Qué información mostró sobre el archivo /etc/passwd? Pruebe este comando con las opciones **-l**, **-c** y **-w** y compare cada una de las tres salidas

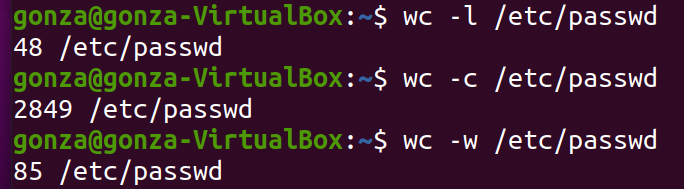
wc /etc/passwd nos dice que el archivo passwd tiene 48 lineas, 85 paabras y 2849 bytes



wc -l /etc/passwd: Nos da la cantidad de líneas

wc -c /etc/passwd: Nos da la cantidad de bytes

wc -w /etc/passwd: Nos da la cantidad de palabras



# Separando (partiendo) un archivo

Para partir un archivo en dos o mas partes de igual tamaño se emplea el comado **split**.

Sintaxis:***split <opciones> <archivo> <prefijo>***

* Ingrese el comando **split --help** y a continuación <ENTER>. ¿Para qué sirven las opciones del comando split?

-a, --suffix-length=N generar sufijos de longitud N (predeterminado 2)

--additional-suffix=SUFFIX agregar un SUFIJO adicional a los nombres de los archivos

-b, --bytes=SIZE poner TAMAÑO bytes por archivo de salida

-C, --line-bytes=SIZE poner como máximo TAMAÑO bytes de registros por archivo de salida

-d use sufijos numéricos que comiencen en 0, no alfabéticos

--numeric-suffixes[=FROM] Lo mismo que -d, pero permite establecer el valor de inicio

-x use sufijos hexadecimales que comiencen en 0, no alfabéticos

--hex-suffixes[=FROM] igual que -x, pero permite establecer el valor de inicio

-e, --elide-empty-files no genera archivos de salida vacíos con '-n'

--filter=COMMAND escribir en la terminal COMANDO; el nombre del archivo es $FILE

-l, --lines=NUMBER poner NÚMERO de líneas/registros por archivo de salida

-n, --number=CHUNKS generar archivos de salida CHUNKS; ver explicación abajo

-t, --separator=SEP utilice SEP en lugar de nueva línea como separador de registros;

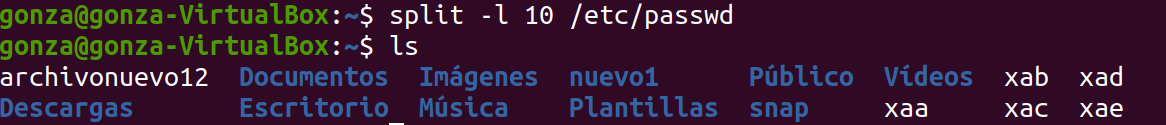
'\0' (cero) especifica el carácter NUL

-u, --unbuffered copie inmediatamente la entrada a la salida con '-n r/...'

--verbose muestra un diagnóstico justo antes de abrir cada

Fichero

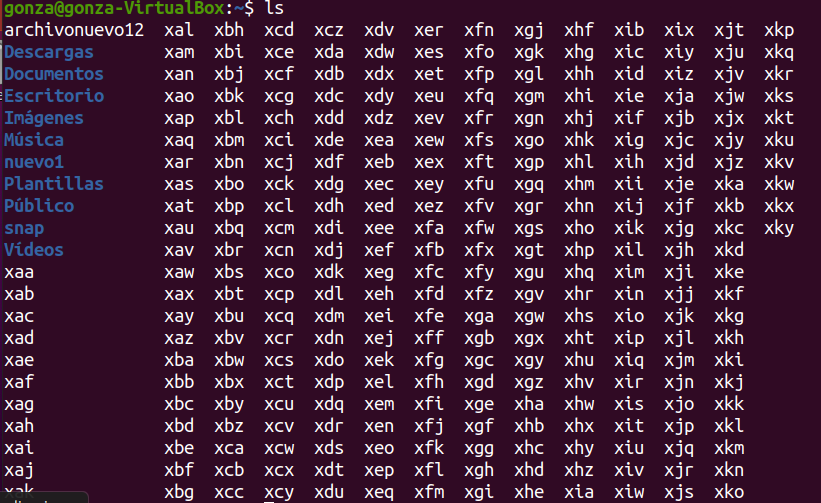
* Ingrese el comando **split -l 10 /etc/passwd partido** y a continuación <ENTER>. ¿Qué haría este comando? Liste los archivos del directorio actual. ¿En cuantas partes separo el archivo password? Determine cuántas líneas tiene cada parte usando el comando wc. Borre todas las partes del archivo **partido**.



Separo el archivo passwd en 5 partes, lo hizo cada 10 líneas de passwd

* Ingrese el comando **split -b 10 /etc/passwd partido** y a continuación <ENTER>.

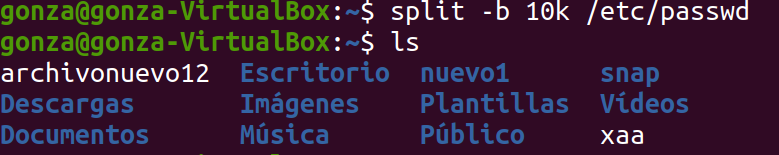
¿Qué haría este comando?Liste los archivos del directorio actual. ¿En cuántas partes separó el archivo password? ¿Son más que las de la opción **-l 10**, por qué? Borre todas las partes del archivo **partido**.



Separo el archivo passwd en 10 bytes cada parte del archivo.

Son mas partes porque con la opción -l, separas por lienas, en cambio aquí separas por tamaño de archivo, al decirle que lo separe solamente por un tamaño de 10, se crearon mas archivos por el tamaño mismo.

* Ingrese el comando **split -b 10k /etc/passwd partido** y a continuación <ENTER>. ¿Qué haría este comando?Liste los archivos del directorio actual. ¿En cuántas partes separó el archivo password? ¿Son más o menos que las de la opción **-b 10**?¿ por qué? Borre todas las partes del archivo **partido**.



Se paro el archivo paswd en 1 parte, esto paso porque se le indico que lo separe en 10 kilobyte, siendo mayor a 10 bytes, por lo que al ser una medida mayo la separo en menos partes

# Uniendo (concatenando) varios archivos

Para concatenar uno o más archivos y visualizarlos en la salida estándar (pantalla) se utiliza el comando **cat**.

Sintaxis: ***cat <opciones> <archivo1> <archivo2> ... <archivon>***

* Ingrese el comando **cat --help** y a continuación <ENTER>. ¿Para qué sirven las opciones del comando cat?

-A, --show-all equivalente a -vET

-b, --number-nonblank número de líneas de salida no vacías, anula -n -e equivalent to -vE

-E, --show-ends mostrar $ al final de cada línea

-n, --number numerar todas las líneas de salida

-s, --squeeze-blank suprimir líneas de salida vacías repetidas

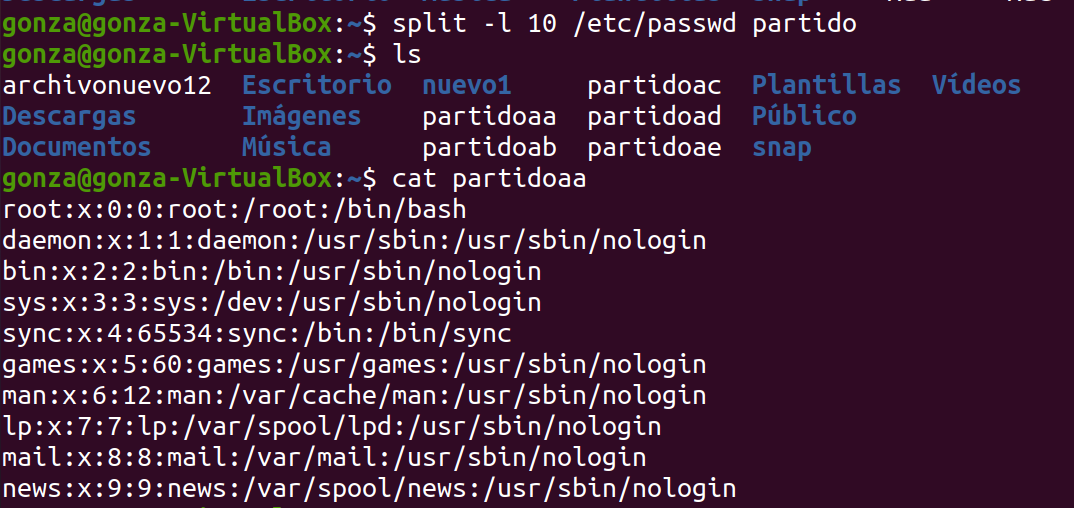
-t equivalente a -vT

-T, --show-tabs muestra los caracteres de tabulación como ^I

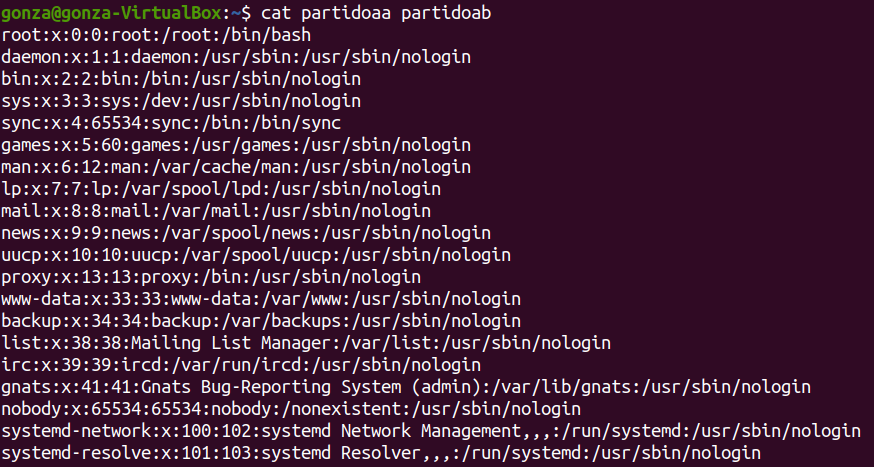
-u (sin efecto)

-v, --show-nonprinting utiliza la notación ^ y M-, salvo para LFD y TAB

* Ingrese el comando **split -l 10 /etc/passwd partido** y a continuación <ENTER>. - Ingrese el comando **cat partidoaa** y a continuación <ENTER>. ¿Que ocurrío? - Ingrese el comando **cat partidoaa partidoab** y a continuación <ENTER>. ¿Y ahora? ¿Nota cómo funciona el comando cat?



Al usar “cat partidoaa” nos mostro el contenido del archivo.



Al usar “cat partidoaa partidoab” nos mostro el contenido del archivo partidoaa y luego el del archivo partidoab

# Otros editores de línea de comando

Existen multitud de editores de línea de comandos para Unix/Linux que podríamos utilizar. El elegir uno en particular dependerá de gustos personales y disponibilidad de los mismos en el sistema que estemos usando.

En particular cabe destacar un conjunto de editores que se crearon con la finalidad de resultar familiares a los usuarios del procesador de textos WordStar para DOS. Estos son:

* Nano.
* Pico.
* Joe.

La disponibilidad de alguno o algunos de éstos dependerá de la distribución de Linux y las opciones de instalación de la misma.

Además existen gran numero de editores orientados a brindar facilidades a los programadores, ejemplo de estos son: Emacs y jed.