**PR\_01.1\_Linux**

*Pablo Checa Blanco*

[Conceptos básicos y variables 1](#_Toc210997771)

[Obtener ayuda y localizar archivos 4](#_Toc1981734300)

[Navegación y listado de archivos 7](#_Toc1985018601)

[Manipulación de Archivos y Directorios 9](#_Toc1518386778)

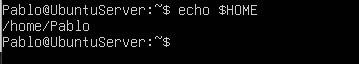
[Archivado y Compresión 12](#_Toc1697887632)

Conceptos básicos y variables

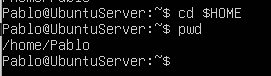
## *Ejercicio 1*

**Muestra el contenido de tu variable de entorno** **HOME . Luego, usa cd junto con esa variable para navegar a dicho directorio y verifica con pwd que te encuentras en la ubicación correcta.**

La variable de entorno HOME representa el directorio personal del usuario que está actualmente conectado.



* echo → Muestra texto o el valor de una variable en la terminal.
* $HOME → El símbolo $ delante del nombre indica que quieres **acceder al valor** de la variable HOME.



* Cd → Change Directory, para navegar entre directorios.
* Pwd → Comprobar el directorio en el que me encuentro

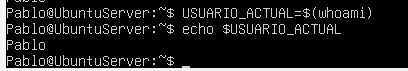
Ejercicio 2

**Ejecuta el comando whoami . Ahora, crea una variable local llamada USUARIO\_ACTUAL que contenga el resultado del comando anterior y muéstrala en la terminal.**



* whoami → Comprobar quien soy, que usuario soy.

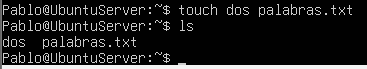
Una **variable local** es una variable que solo existe **en la terminal actual** (en la sesión de shell). A la terminal, se borra.



* USUARIO\_ACTUAL= → Se crea una variable llamada USUARIO\_ACTUAL.
* $(whoami) → Es una forma de **capturar el resultado de un comando**. Esto se llama **expansión de comandos**.

## *Ejercicio 3*

**Intenta crear un archivo llamado dos palabras.txt sin usar comillas. Observa el resultado con ls . ¿Qué ha ocurrido y por qué? Ahora, bórralo(s) y créalo correctamente.**



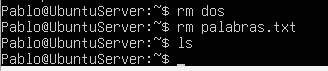
* touch → crear archivo vacio
* Ls → ver archivos

Al no usar comillas alrededor de dos palabras.txt, el shell **interpreta los espacios como separador de argumentos**. Esto se interpreta como si se creasen **dos archivos diferentes.**

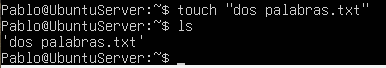
* **rm dos palabras.txt**

Esto volverá a fallar por el mismo motivo: el shell piensa que le estás pasando **dos argumentos**. Para evitarlo, debemos de ponerlo de alguna de estas dos formas:

* Rm dos y rm palabras.txt
* Rm “dos” “palabras.txt”

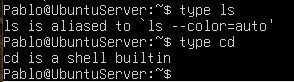


Y ahora lo creamos correctamente:



## *Ejercicio 4*

**Usa el comando type para averiguar si ls y cd son internos o externos al shell. ¿Qué diferencia práctica crees que implica esto?**



**Comando interno:** Forma parte del propio shell, no necesita buscarse en disco.

**Comando externo:** Es un programa independiente, que vive en el sistema de archivos.

La clave para diferenciarlos es saber si modifica el entorno del shell o no:

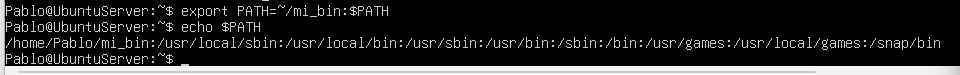
* cd (cambiar de directorio) **debe modificar el entorno del shell actual**, y **solo puede hacerlo si está integrado** dentro del shell.
* Si fuera externo, cambiaría el directorio en un proceso hijo y no tendría efecto en tu terminal.
* ls (listar archivos) **no modifica el entorno del shell**, simplemente muestra información. Por eso puede estar implementado como un programa aparte (en /bin/ls).

## *Ejercicio 5*

Muestra tu PATH actual. Crea un directorio ~/mi\_bin y añádelo temporalmente al principio de tu PATH . Verifica que el cambio se ha realizado correctamente.



* mkdir crea un directorio.
* ~ es una forma rápida de referirse a tu **directorio personal** (/home/tu\_usuario).

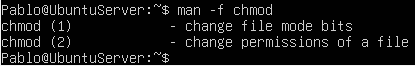
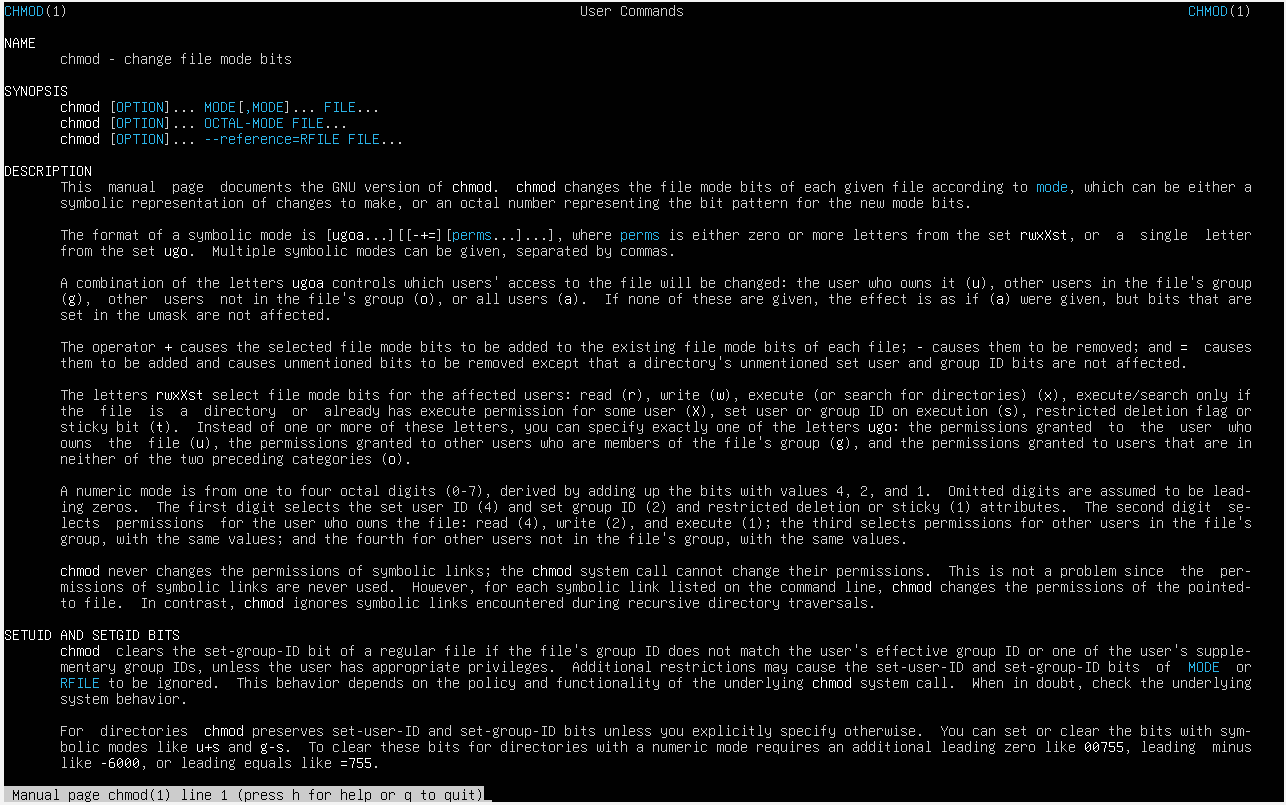


* export actualiza la variable de entorno para que esté disponible en los procesos hijos.
* Estamos asignando a PATH un nuevo valor que es:  
   **~/mi\_bin + : + valor actual de PATH**.
* Esto hace que al buscar comandos, el shell busque primero en ~/mi\_bin, luego en los otros directorios.

Obtener ayuda y localizar archivos

Ejercicio 1

Abre la página del manual para el comando chmod . ¿En qué sección del manual se encuentra? ¿Qué indica ese número de sección sobre el tipo de comando?



La página del manual de chmod está en la sección 1. (Se vé arriba a la izquierda)

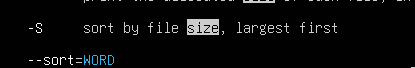
Esto indica que chmod es un comando de usuario, es decir, un programa ejecutable que puedes usar en la terminal.

Los comandos en la sección 1 son los que normalmente usas para administrar y manipular el sistema como usuario normal.

Ejercicio 2

Usando la función de búsqueda dentro de la página del manual de ls , encuentra la opción que ordena los archivos por tamaño.

Abrimos el manual con el comando man ls y para buscar pulsamos “/” y escribimos “size”, es similar a Ctrl+F en un PDF. Para navegar entre los resultados, pulsamos n y N.



Ahora que sabemos que hay que utilizar –S, salimos del manual y lo comprobamos:



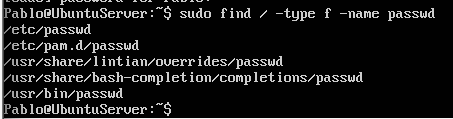
Ejercicio 3

Imagina que has olvidado dónde se guarda el archivo de configuración de usuarios. Sabiendo que se llama passwd , usa find para buscarlo desde el directorio raíz ( / ). Anota la ruta completa que has encontrado.

El comando sería:

sudo find / -type f -name passwd

* sudo – permite buscar en todo el sistema, incluso en carpetas protegidas. -
* find – es el comando para buscar archivos.
* / – indica que la búsqueda empieza en el directorio raíz (todo el sistema).
* -type f – busca solo archivos (no carpetas).
* -name passwd – busca archivos con el nombre exacto passwd.



Ejercicio 4

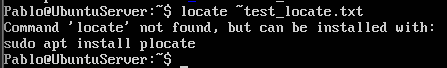
Crea un archivo vacío llamado test\_locate.txt en tu directorio home. Inmediatamente después, búscalo con locate . ¿Aparece en los resultados? ¿Por qué sí o por qué no?

Crear el archivo:

Home es “~”



Ahora al utilizar el comando locate nos pide instalarlo:





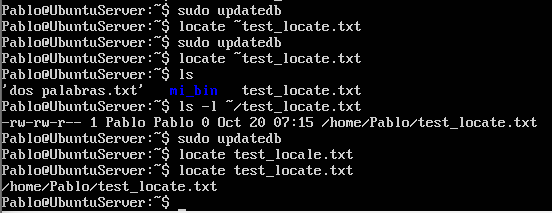


A mí no me ha aparecido una vez ejecutado el comando, se debe a que el comando locate usa una base de datos indexada que no se actualiza en tiempo real. Para que apareciese habría que actualizar con un comando la base de datos y posteriormente volver a ejecutar el comando.

Ejercicio 5

Basado en el ejercicio anterior, ¿qué comando (probablemente con sudo ) necesitas ejecutar para que locate sí encuentre tu archivo? Ejecútalo y verifica que ahora sí lo encuentras.

Hay que actualizar la BBDD con el comando: sudo updatedb

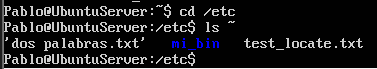


Al principio no me encontraba el archivo a pesar de actualizar la BBDD, por lo qué verifiqué que realmente el archivo existiese, luego volví a actualizar la BBDD y finalmente volví a localizarlo y lo encontró.

Navegación y listado de archivos

Ejercicio 1

Navega al directorio /etc . Desde ahí, sin usar cd , lista el contenido de tu directorio home usando una ruta con el atajo ~ .



“~” Representa siempre el directorio home sin importar donde estemos ubicados.

Ejercicio 2

Desde tu directorio home , navega a / y luego a var y finalmente a log usando una sola línea de comando y rutas relativas.



Rutas relativas: Son rutas que se escriben en relación a donde estamos ubicados, que no empiezan por /. “..” Volvemos al directorio de atrás ‘al padre’ ” y “.” significa en el directorio actual. Entonces, con el primer .. volvemos a home y con el segundo a “/” que es la raíz. Luego ya podemos agregar /var/log, que es a donde queremos llegar.

Ejercicio 3

Lista el contenido de /etc en formato largo. En la salida, identifica el propietario, el grupo y los permisos del archivo passwd .

Hay que ejecutar este comando: ls -l /etc, pero como la salida es muy larga, para poder ver todo y navegar en ella utilizaremos: ls -l /etc | less. Pudiendo así utilizar las flechas para navegar entre los resultados.



Interpretar esa línea correctamente:

* El primer caracter “-” indica que es un archivo regular.
* En los grupos de tres letras:

- rw = Permisos del propietario (Read and Write)

- r-- = permisos del grupo (Solo Read)

- r-- = permisos para otros usuarios: (Solo Read)

* 1º Root: Propietario
* 2º Root: Grupo
* 1688 = Tamaño del archivo en bytes
* “Fecha” = Última fecha de modificación

Entonces: El propietario es root, el grupo también es root y los permisos son lectura y escritura para el propietario, mientras que es solo lectura para el grupo y otros usuarios.

Ejercicio 4

Compara la salida de ls -l /etc y ls -lh /etc . ¿Qué hace la opción h y por qué es útil para las personas?

La opción h humaniza los resultados, es decir, que para el humano sea más fácil calcular el espacio que ocupa el archivo. Voy a poner el ejemplo del ejercicio 3:

Sin h:



Con h:

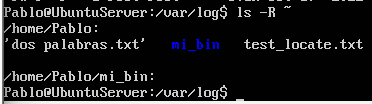


Se puede observar que el formato del tamaño cambia, pudiendo saber perfectamente que estos archivos ocupan 1,7 Kilobytes.

Ejercicio 5

Ejecuta ls -R ~ . ¿Qué hace la opción R ? ¿Por qué podría ser peligroso usarla en el directorio raíz ( / )?

Este comando muestra recursivamente “de ahí -R” todos los archivos y subdirectorios dentro del directorio personal (~).



Si probamos a utilizar “ls -R /” estamos diciendo que liste recursivamente todo el contenido del sistema, empezando desde la raíz /. Lo cual supondría varios problemas principalmente por su tamaño y complejidad:

* Enorme cantidad de datos.
* El rendimiento de la CPU y el Disco puede verse afectado
* Hay archivos especiales que pueden comportarse de forma especial al listarlos.

Manipulación de Archivos y Directorios

Ejercicio 1

Crea la estructura de directorios proyecto/src , proyecto/doc y proyecto/bin usando un único comando mkdir .

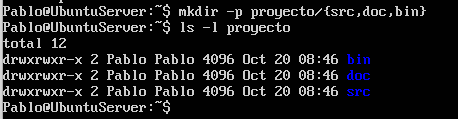
Comando: mkdir -p proyecto/{src,doc,bin}

Qué hace cada parte:

mkdir: Es el comando para crear directorios.

-p: Significa "padres" o "crear padres si no existen".  
 Esto hace que si la carpeta proyecto no existe, la cree automáticamente antes de crear las subcarpetas.

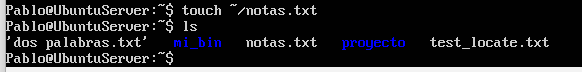
proyecto/{src,doc,bin}: Esto usa expansión de llaves en la terminal. La terminal interpreta esto como tres rutas diferentes: proyecto/src, proyecto/doc y proyecto/bin



Ejercicio 2

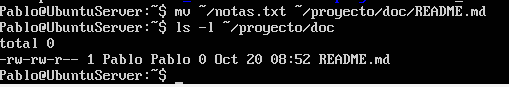
Crea un archivo ~/notas.txt . Muévelo a ~/proyecto/doc y, en el mismo comando, renómbralo a README.md .

Primero creamos el archivo:



* El comando touch crea un archivo vacío (o actualiza la fecha de modificación si ya existe).

Ahora lo movemos y lo renombramos en una misma linea:



* mv sirve para mover archivos o carpetas.
* El primer argumento es la ruta del archivo a mover (~/notas.txt).
* El segundo argumento es la ruta destino, si cambiamos el nombre en esa ruta, el archivo se renombra al mismo tiempo (~/proyecto/doc/README.md).

Ejercicio 3

Copia el archivo README.md de proyecto/doc a proyecto/bin . Luego, borra el archivo original de la carpeta doc .



* Esto copia el archivo desde proyecto/doc a proyecto/bin



* Esto elimina el archibo original que quedó en /doc

Ejercicio 4

Intenta borrar el directorio proyecto con rmdir . ¿Qué error obtienes? Ahora, usa rm con la opción correcta para borrar el directorio y todo lo que contiene.

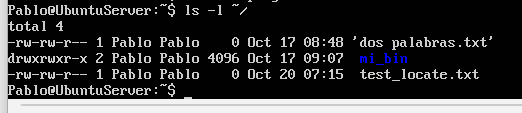


El error dice básicamente que no se puede borrar porque el directorio no está vacío, y “rmdir” solo borra directorios vacíos.

Utilizamos ahora rm:



Comprobamos:



Ejercicio 5

Navega a /etc . Usando un solo comando ls con globbing , lista todos los archivos que empiecen con la letra s y terminen con .conf .



* S\* -> que empiecen por s
* .conf -> que termine exactamente con conf

Archivado y Compresión (EJ 4 MIRAR)

Ejercicio 1

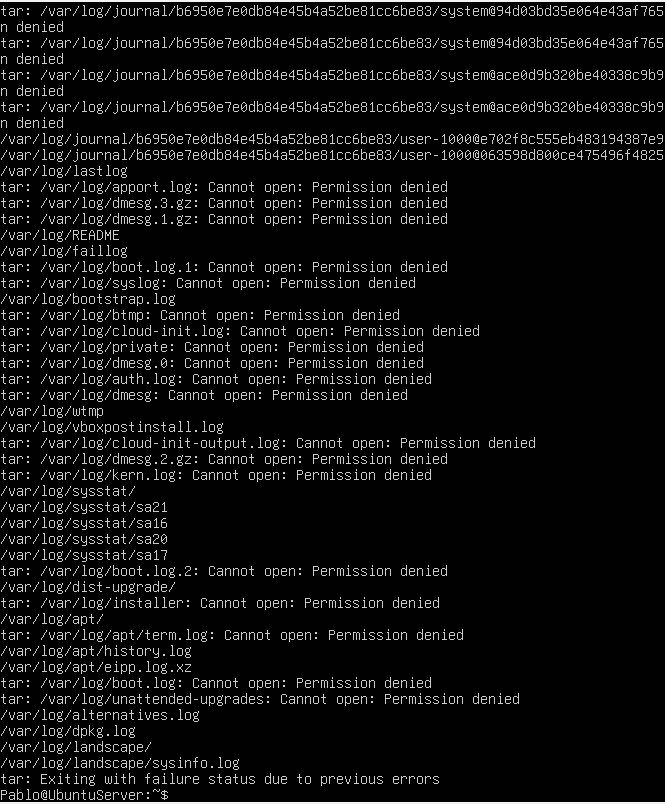
Crea un archivo tar llamado log\_backup.tar que contenga todos los archivos del directorio /var/log . ¿Qué advertencias de “permiso denegado” aparecen y por qué?

El comando a utilizar es:

* tar -cvf log\_backup.tar /var/log

Explicación de cada parámetro:

* tar: comando para empaquetar o extraer archivos.
* -c: crea un nuevo archivo tar.
* -v: modo verbose (muestra los archivos que va procesando).
* -f log\_backup.tar: especifica el nombre del archivo de salida.
* /var/log: carpeta que queremos empaquetar.



Los permisos denegados ocurren debido a que existen archivos en /var/log que solo pueden ser leidos por el usuario root.

Ejercicio 2

Comprime el archivo log\_backup.tar con gzip . Compara el tamaño del archivo original y el comprimido usando ls -lh .

Comprobamos que estamos bien ubicados:

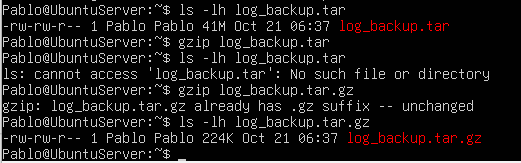


* gzip: comando para comprimir archivos.
* Este comando reemplaza el archivo original (log\_backup.tar) por su versión comprimida (log\_backup.tar.gz).
* No deja una copia del original a menos que uses -k (para “keep”).

Miramos primero el tamaño del original:



Ahora lo comprimimos y volvemos a mirar:

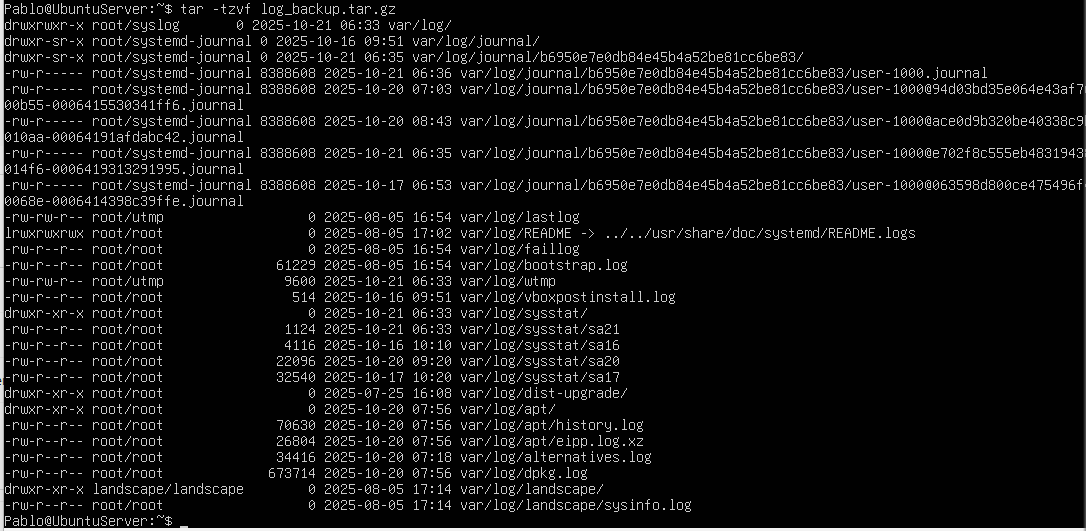


Ejercicio 3

Lista el contenido del archivo log\_backup.tar.gz sin extraerlo para verificar que los archivos están dentro.

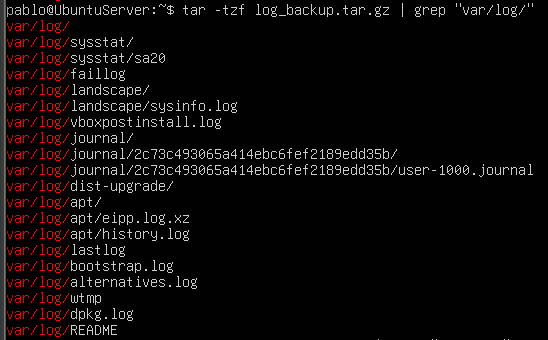
El comando es: tar -tzvf log\_backup.tar.gz

* T -> listar el contenido del archivo tar
* Z -> indica que el archivo está comprimido con gzip
* V -> muestra los archivos uno por uno
* F log... especifica el nombre del archivo



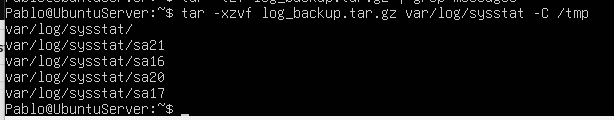
Ejercicio 4

Extrae únicamente el archivo syslog (o messages ) de log\_backup.tar.gz a tu directorio /tmp .





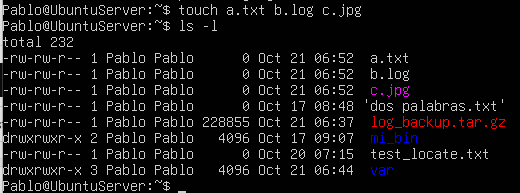
No existe ningun archivo syslog o messages, por lo que voy a probar a extraer por ejemplo el sysstat, para poder realizar el ejercicio.



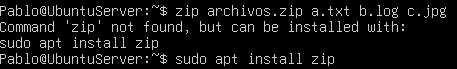
Ejercicio 5

Crea tres archivos ( a.txt , b.log , c.jpg ) y luego crea un archivo zip que los contenga.

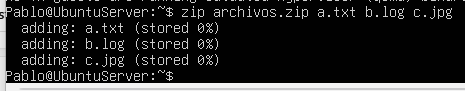
Creamos los archivos:



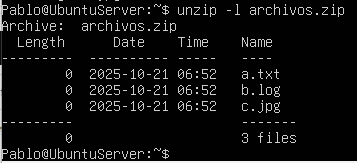
Descargamos zip



Una vez descargado, volvemos a ejecutar el comando:

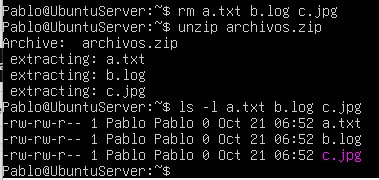


Comprobamos el zip:



Ejercicio 6

Elimina los tres archivos originales y luego recupéralos desde el archivo zip .



Como apunte, si quisieramos recuperar solo un archivo en concreto, especificariamos el nombre del archivo en el comando:

* unzip archivos.zip b.log

Ejercicio 7

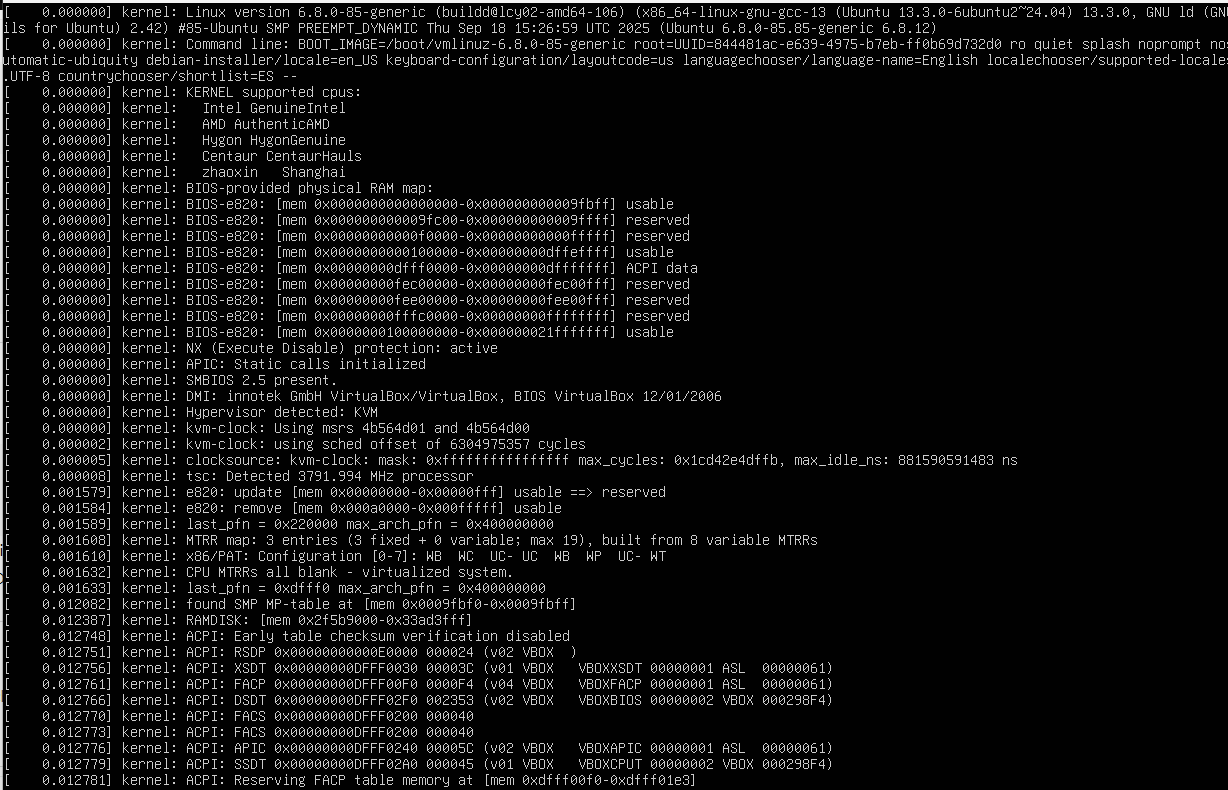
Usa zcat (o gzcat ) para leer el contenido de un archivo de log comprimido (ej: en /var/log , busca uno que termine en .gz ) sin crear un archivo descomprimido.

Buscamos:



Utilizamos el comando con less para poder navegar:

zcat /var/log/dmesg.1.gz | less



Redirección, tuberías y filtros

Ejercicio 1

Guarda la lista de archivos de tu directorio home (formato largo) en un archivo mis\_archivos.txt .



Ejercicio 2

Sin borrar el contenido anterior, añade la fecha y hora actual al final del archivo mis\_archivos.txt .



Ejercicio 3

Usa grep y una tubería ( | ) para contar el número de directorios que hay en /etc . (Pista: ls -l | grep '^d' ).

El comando completo sería:

ls -l /etc | grep '^d' | wc –l

En donde:

* ls -l /etc — Lista el contenido de /etc con detalles.
* Cada línea que representa un directorio comienza con una d en el primer carácter (permiso).
* grep '^d' — Filtra solo las líneas que empiezan con d (directorios).
* Para contar las líneas que quedan (es decir, el número de directorios), usamos wc -l.



106 directorios.

Ejercicio 4

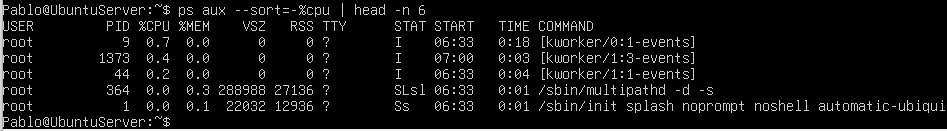
Muestra las 10 últimas líneas del archivo /etc/passwd y, usando otra tubería, extrae solo los nombres de usuario (el primer campo).



* tail -n 10 /etc/passwd: Muestra las 10 ultimas lineas del archivo
* | (tubería)
* Cut –d':’ -f1 Extrae el primer campo (-f1) usando : como delimitador (-d’:’) que corresponde al nombre de usuario.

Ejercicio 5

Muestra una lista de todos los procesos del sistema ( ps aux ), ordénala por uso de CPU (tercera columna) y muestra solo las 5 líneas superiores.

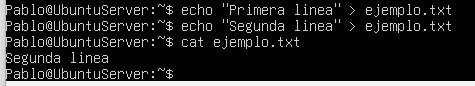


* Ps aux: lista todos los procesos del sistema con detalles
* --sort=-%cpu: ordena los procesos por la columna %CPU de mayor a menor (El signo – quiere decir que vaya descendente)
* Head –n 6 Muestra las primeras 6 lineas (La primera es el encabezado y las 5 siguientes son los procesos mas consumistas)

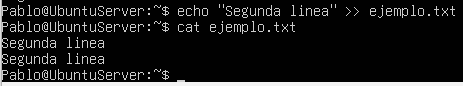
Ejercicio 6

¿Cuál es la diferencia entre usar > y >> para redirigir la salida de un comando a un archivo? Demuéstralo con un ejemplo.

>: Redirección que sobrescribe: crea un archivo nuevo o borra el contenido existente y escribe la salida del comando.



>>: Redirección que añade: agrega la salida al final del archivo, sin borrar el contenido previo.



Ejercicio 7

Ejecuta find /etc -name "\*.conf" . Redirige la salida estándar a un archivo config\_files.txt y los errores (si los hay) a errors.txt .



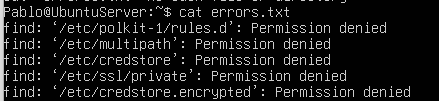
* find /etc -name "\*.conf" Busca archivos con extensión .conf en /etc
* > Redirige la salida estándar al archivo config\_files.txt
* 2> Redirige la salida de errores (Con descriptor nº2) al archivo errors.txt

Comprobar archivos encontrados:

* Cat config\_files.txt



Ver si hubo errores:



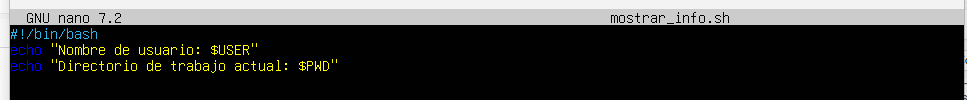
Errores de permisos.

Scripts básicos (MIRAR EJERCICIO 6)

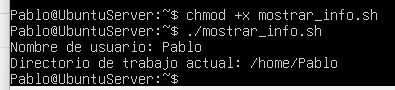
Ejercicio 1

Crea un script que imprima tu nombre de usuario y el directorio de trabajo actual usando las variables de entorno correspondientes

Creamos el script:



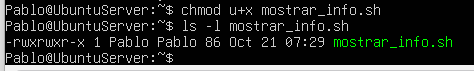
Dar permisos de ejecución al script y mostrarlo:



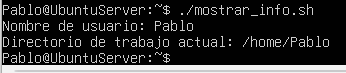
Ejercicio 2

Haz el script anterior ejecutable solo para ti ( chmod u+x ... ) y ejecútalo. Luego, intenta ejecutarlo como otro usuario (si es posible) o explica qué pasaría.

Ponemos permisos solo para mi:



Volvemos a ejecutar:

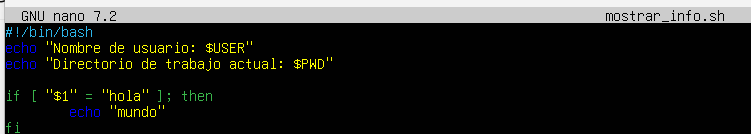


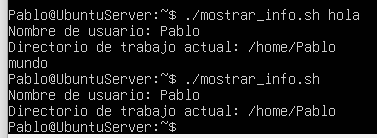
Si otro usuario ejecuta el comando recibirá un error de Permiso denegado, porque el permiso de ejecución solo está para el propietario (u+x).

Ejercicio 3

Modifica el script para que acepte un argumento. Si el argumento es “hola”, debe imprimir “mundo”. Si es cualquier otra cosa, no debe imprimir nada.

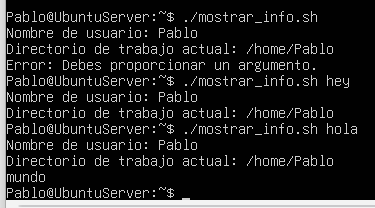
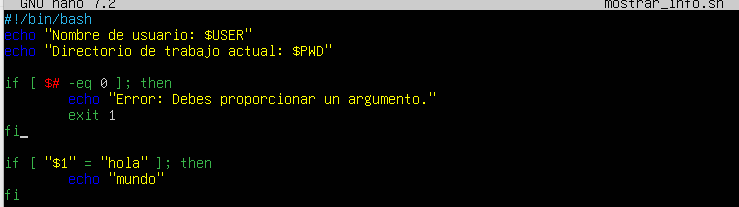
Modificamos el archivo:

“$1” es el primer argumento que se le pasa al script.



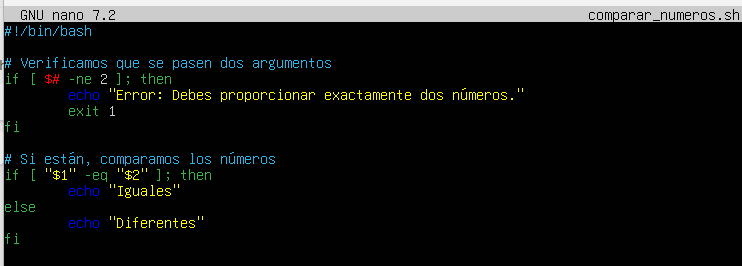
Ejercicio 4

Mejora el script anterior para que, si no se proporciona ningún argumento, muestre un mensaje de uso: “Error: Debes proporcionar un argumento.”

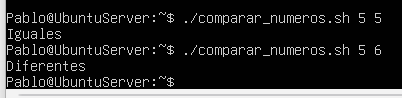


Ejercicio 5

Escribe un script que reciba dos números. Debe imprimir “iguales” si son iguales y “diferentes” si no lo son.



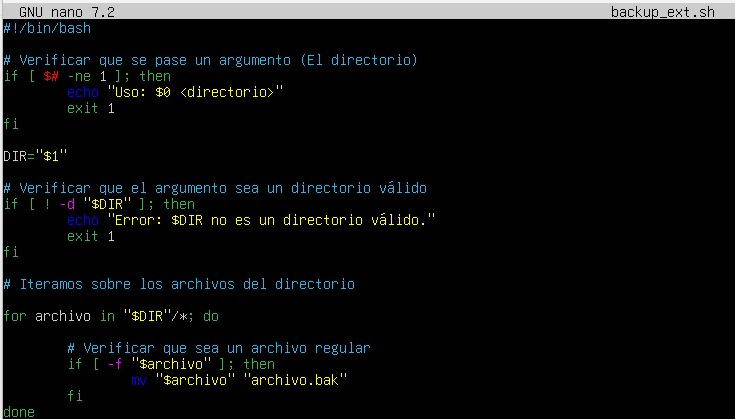
Damos los permisos de ejecución y probamos:





Ejercicio 6

Escribe un script que, dado un directorio como argumento, use un bucle for para iterar sobre su contenido ( ls $1 ) y añada la extensión .bak a cada archivo.



Ya añadió

