Comandos

clear

* solo borra el contenido visual en la ventana de la terminal, pero no afecta a los archivos ni directorios en el sistema. Sin embargo, existen otros comandos relacionados con la gestión y el mantenimiento de la terminal que también son útiles y que no eliminan datos del sistema.

reset:

* El comando reset se utiliza para restablecer la terminal a su estado predeterminado, limpiando la pantalla y restaurando las configuraciones por defecto del terminal. Puede ser útil si la terminal se ha vuelto incoherente o se ha desconfigurado de alguna manera.

tput:

* El comando tput se utiliza para controlar las características de la terminal, como cambiar colores, mover el cursor y borrar líneas. Puede ser útil para realizar tareas de formato en la salida de comandos o scripts.
  + Ejemplo: tput clear realiza una acción similar a clear.

stty (Set Terminal Type):

* El comando stty se utiliza para configurar y mostrar las características de la terminal, como el tamaño del terminal, la velocidad de transmisión, el modo de control y más. Puede ser útil para personalizar la configuración de la terminal.
  + Ejemplo: stty -icanon cambia el modo de entrada de la terminal.

resize:

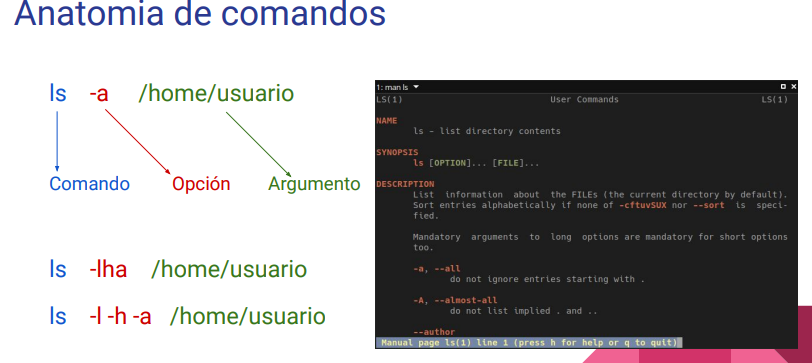
* El comando resize se utiliza para cambiar el tamaño de la ventana de la terminal. Puede ser útil si necesitas ajustar manualmente el tamaño de la terminal.
  + Ejemplo: resize -s 40 80 cambia el tamaño de la terminal a 40 filas y 80 columnas.

Otros comandos

* history: se utiliza para mostrar una lista de los comandos que has ejecutado en la terminal durante la sesión actual.
* pwd: Muestra el directorio de trabajo actual.
* more / less: Permite ver archivos de texto página por página.
* head / tail: Muestra las primeras o últimas líneas de un archivo.
* grep: Busca patrones de texto en archivos.
* find: Busca archivos y directorios en el sistema.
* chmod: Cambia los permisos de archivo.
* chown: Cambia el propietario de un archivo o directorio.
* ps: Muestra información sobre los procesos en ejecución.
* kill: Envía señales a procesos para detenerlos o administrarlos.
* top: Muestra información en tiempo real sobre el uso de recursos del sistema.
* df: Muestra el espacio en disco disponible.
* du: Muestra el uso de espacio en disco de archivos y directorios.
* tar: Gestiona archivos y compresión.
* ssh: Inicia una sesión segura de shell remoto.
* scp: Copia archivos de forma segura entre sistemas.
* ping: Verifica la conectividad de red.
* ifconfig / ip: Muestra y configura la información de red.

----------------------------------------------------------------------------------------------------------------

BASH 101 - 1



El comando ls se utiliza para listar los archivos y directorios en el directorio actual. Puede utilizarse con diferentes opciones para mostrar información adicional, como tamaños de archivo, permisos y fechas de modificación. Algunos ejemplos de uso comunes son:

* ls: Lista los archivos y directorios en el directorio actual.
* ls -l: Lista los archivos y directorios en formato largo, mostrando detalles como permisos, propietario, grupo, tamaño y fecha de modificación.
* ls -a: Lista todos los archivos, incluyendo los ocultos (que comienzan con un punto)
* ls directorio: Lista los archivos y directorios en un directorio específico.
* file:El comando file se utiliza para determinar el tipo de archivo de un archivo dado. Puede identificar archivos binarios y mostrar información sobre el tipo de archivo, como texto, imágenes, archivos ejecutables, etc. Para usarlo, simplemente proporciona el nombre del archivo como argumento.
* pwd (Print Working Directory): Este comando muestra el directorio actual en el que te encuentras. Es útil para saber en qué parte del sistema de archivos te encuentras en un momento dado.

cd <directorio>: Este comando se usa para cambiar al directorio especificado. Por ejemplo, si quieres cambiar al directorio llamado "documentos", puedes usar cd documentos.(cd documentos), si deseas moverte a un directorio llamado "fotos" que está en el mismo nivel que tu directorio actual, puedes hacerlo así: cd fotos

* cd .. : Este comando se usa para moverte un nivel hacia arriba en la estructura de directorios. Por ejemplo, si estás en "/home/usuario/documentos" y ejecutas cd .., te moverás a "/home/usuario".
* cd: Si ejecutas cd sin ningún argumento, te llevará de vuelta a tu directorio personal. Esto es útil cuando estás navegando por varios directorios y quieres regresar rápidamente a tu directorio de inicio.
* cd -: Este comando te llevará de vuelta al directorio anterior en el que estabas antes de moverte al directorio actual. Es útil para alternar entre dos directorios de manera eficiente. Por ejemplo, si estabas en "/home/usuario/documentos" y luego te moviste a "/var/archivos" usando cd /var/archivos, puedes regresar a "/home/usuario/documentos" ejecutando cd -.

--------------------------------------------------------------------------------------------------------------

BASH 101 - 2

Manipulación

* mv (Move): Este comando se utiliza para mover o renombrar archivos y directorios. Puedes cambiar la ubicación de un archivo o directorio al proporcionar el nombre del archivo o directorio y la nueva ubicación. También puedes utilizarlo para cambiar el nombre de un archivo o directorio al proporcionar el nombre actual y el nuevo nombre. Ejemplo de mover un archivo: mv archivo.txt /ruta/nueva/ . Ejemplo de cambiar el nombre de un archivo: mv archivo\_viejo.txt archivo\_nuevo.txt
* cp (Copy):El comando cp se utiliza para copiar archivos o directorios. Puedes copiar un archivo de una ubicación a otra o hacer copias de un archivo en diferentes ubicaciones. Ejemplo de copiar un archivo: cp archivo\_origen.txt /ruta/destino/
* rm (Remove): Este comando se utiliza para eliminar archivos o directorios. Ten cuidado al usarlo, ya que los archivos eliminados no se pueden recuperar fácilmente. Ejemplo de eliminar un archivo: rm archivo.txt
* mkdir (Make Directory): El comando mkdir se utiliza para crear un nuevo directorio o carpeta en la ubicación especificada. Ejemplo de crear un directorio: mkdir nuevo\_directorio
* touch: El comando touch se utiliza para crear un nuevo archivo vacío o actualizar la fecha de modificación de un archivo existente. Ejemplo de crear un nuevo archivo: touch nuevo\_archivo.txt
* cat (Concatenate): Este comando se utiliza para mostrar el contenido de un archivo en la línea de comandos. Ejemplo de mostrar el contenido de un archivo: cat archivo.txt

Esto mostrará todo el contenido del archivo en la terminal.

* tail: El comando tail se utiliza para mostrar las últimas líneas de un archivo de texto. Puedes especificar cuántas líneas quieres ver como argumento. Ejemplo de mostrar las últimas 10 líneas de un archivo: tail -n 10 registro.txt Si deseas ver las últimas líneas en tiempo real a medida que se agregan al archivo (como un registro en tiempo real o un archivo de registro en crecimiento), puedes usar la opción -f para seguir el archivo: tail -f registro.txt

De esta manera, el comando seguirá mostrando las nuevas líneas a medida que se agregan al archivo "registro.txt". Puedes detenerlo presionando Ctrl + C cuando desees detener la visualización en tiempo real.

---------------------------------------------------------------------------------------------

Usar less: less es un visor de texto interactivo que te permite desplazarte hacia arriba y hacia abajo a través del contenido del archivo. Puedes usarlo de la siguiente manera: less archivo.txt

* Para avanzar hacia adelante, simplemente presiona la tecla "Espacio". Para retroceder, presiona "B". Para salir de less, presiona la tecla "q".
* Para avanzar hacia adelante, presiona la tecla "Espacio". Para retroceder, presiona "B". Para salir de more, presiona la tecla "q".

Usar un editor de texto (como nano o vim): Si deseas editar el archivo mientras lo ves o si necesitas realizar cambios en él, puedes utilizar un editor de texto como nano o vim. Por ejemplo, para abrir "archivo.txt" con nano: nano archivo.txt

Para abrirlo con vim: vim archivo.txt

* Estos editores te permitirán ver y editar el contenido del archivo. Asegúrate de estar familiarizado con los comandos de navegación y edición del editor que elijas. Para salir de nano, generalmente presiona Ctrl + X, seguido de Y para confirmar los cambios si los has realizado. Para salir de vim, presiona Esc, luego escribe :q (:q! forzado) y presiona Enter.

Ejemplo : vim Desktop/asd.txt

VI/VIM

* vi archivo ó vim archivo
* i -> entra en modo de edición
* esc -> sale del modo actual
* : -> insertar comandos
* :w -> grabar archivo
* :q -> sale de vim
* :q! -> sale sin guardar
* :wq -> graba y sale
* /texto -> busca patron
* :%s/a/b/g -> reemplaza las a por b

Como usar nano

Para guardar un archivo en Nano, sigue estos pasos:

1. Asegúrate de que estás en el archivo que deseas guardar o editando el archivo correcto en Nano.
2. Si has realizado cambios en el archivo, puedes confirmar esto observando un asterisco (\*) en el extremo inferior izquierdo de la pantalla de Nano, lo que indica que el archivo ha sido modificado.
3. Para guardar los cambios en el archivo, presiona la combinación de teclas **Ctrl** + **O**. Esto abrirá la opción "Write Out" en la parte inferior de la pantalla.
4. Luego, presiona **Enter** para confirmar que deseas guardar el archivo. Si deseas guardar el archivo con un nombre diferente o en una ubicación diferente, puedes editar el nombre del archivo antes de presionar **Enter**.
5. Nano guardará el archivo y mostrará un mensaje en la parte inferior de la pantalla confirmando que el archivo se ha guardado.
6. Finalmente, puedes salir de Nano presionando **Ctrl** + **X**.

Recuerda que si estás editando un archivo que requiere permisos de administrador (por ejemplo, un archivo en el directorio **/etc**), es posible que debas ejecutar Nano con privilegios de superusuario utilizando **sudo** para poder guardar los cambios. Por ejemplo:

Copy code

sudo nano nombre\_del\_archivo

Después de hacer esto, los pasos anteriores para guardar el archivo seguirán siendo los mismos.

-----------------------------------------------------------------------------------------

Comandos Fundamentales – 1

man commando:

* El comando man se utiliza para mostrar el manual de un comando específico. Sustituye "commando" por el nombre del comando del que deseas obtener información. Por ejemplo, para ver el manual del comando "ls", puedes ejecutar: man ls

man:

* Este comando muestra el manual del propio comando man. Puede ser útil para aprender cómo utilizar el comando man y acceder a la documentación del sistema.

man 2 fork:

* + Este comando man se usa para acceder a la documentación de la llamada de sistema fork. El número "2" en "man 2" indica que estás buscando información sobre llamadas al sistema.

man 3 scanf:

* Este comando man se usa para acceder a la documentación de la función de biblioteca estándar C scanf. El número "3" en "man 3" indica que estás buscando información sobre funciones de biblioteca estándar.

which:

* El comando which se utiliza para mostrar la ruta completa de un comando. Si proporcionas el nombre de un comando como argumento, which te mostrará la ubicación del ejecutable del comando en el sistema. Por ejemplo: which ls

apropos:

* El comando apropos se utiliza para buscar comandos o funciones relacionados con una palabra clave o término en las descripciones de los comandos y las páginas del manual. Por ejemplo, si deseas buscar comandos relacionados con "archivos", puedes hacerlo de la siguiente manera: apropos archivos

file:

* El comando file se utiliza para obtener información sobre el tipo de archivo. Puedes proporcionar el nombre de un archivo como argumento y file te dirá el tipo de archivo, como texto, imagen, ejecutable, etc. Por ejemplo: file archivo.txt

------------------------------------------------------------------------------

Comandos Fundamentales – 2

Manipulación de archivos:

touch:

* Este comando se utiliza para crear un archivo vacío con el nombre especificado. Por ejemplo, para crear un archivo llamado "nuevo\_archivo.txt", puedes ejecutar: touch nuevo\_archivo.txt

Ejemplo

touch programa.h

touch programa.hs

touch programa.txt

rm:

* El comando rm se utiliza para eliminar archivos o directorios. Ten cuidado al usarlo, ya que los archivos eliminados no se pueden recuperar fácilmente. Puedes usar -r para eliminar directorios de manera recursiva y -f para forzar la eliminación sin confirmación.
  + - Eliminar un archivo: rm archivo.txt
    - Eliminar un directorio y su contenido: rm -r directorio

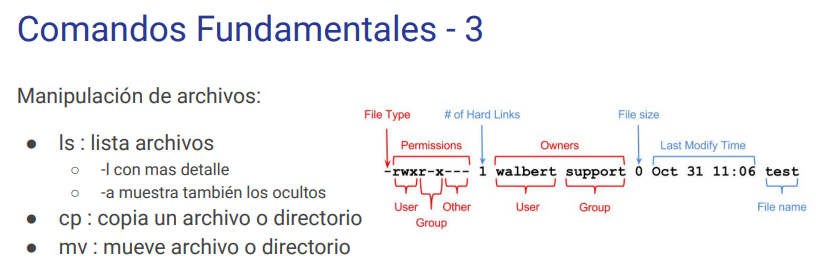
ln:

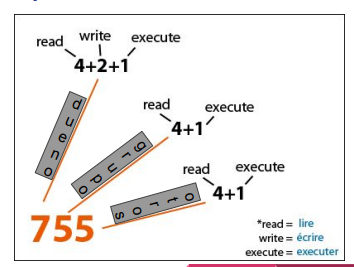
* El comando ln se utiliza para crear enlaces (links) entre archivos. Puede ser utilizado de dos formas principales:
  + - ln old new: Crea un enlace llamado "new" que apunta al mismo contenido que "old". Esto crea un enlace duro.
    - ln -s old link: Crea un enlace simbólico (también conocido como symlink o acceso directo) llamado "link" que apunta a "old". Los enlaces simbólicos son como atajos a otros archivos o directorios.

mkdir:

* El comando mkdir se utiliza para crear directorios o carpetas. Puedes usar la opción -p para crear directorios en una jerarquía completa, incluyendo los directorios padres si no existen.
  + - Crear un directorio: mkdir nuevo\_directorio
    - Crear una jerarquía completa de directorios: mkdir -p un/arbol/de/directorios

---------------------------------------------------------------------------------------------------





Comandos Fundamentales - 4

du (Disk Usage):

* El comando du se utiliza para calcular y mostrar el tamaño en disco de un directorio y sus subdirectorios. Algunas opciones comunes incluyen:
  + - -h: Muestra los tamaños en un formato legible por humanos, como "KB", "MB" o "GB".
    - -s: Muestra solo el tamaño total del directorio, sin detalles de subdirectorios.
    - -a | sort -n: Muestra el tamaño de todos los archivos y directorios en el directorio actual y los ordena de manera ascendente por tamaño.

Ejemplo para mostrar el tamaño de un directorio de forma legible por humanos: du -h directorio

df (Disk Free):

* El comando df se utiliza para mostrar el uso de los discos en el sistema. Proporciona información sobre la cantidad de espacio disponible y utilizado en cada sistema de archivos montado en el sistema.
  + df -h

quota:

* El comando quota se utiliza para ver la utilización de la cuota de disco de un usuario en sistemas con cuotas de disco habilitadas. Proporciona información sobre la cantidad de espacio utilizado y permitido para ese usuario.

quota -u usuario

chmod (Change Mode):

* El comando chmod se utiliza para cambiar los permisos de archivos o directorios. Puedes cambiar los permisos utilizando notaciones numéricas (como 664) o notaciones simbólicas (como +x o -r). Algunos ejemplos:
  + - chmod +x script.sh: Agrega permisos de ejecución al archivo "script.sh".
    - chmod 664 archivo: Establece permisos de lectura y escritura para el propietario y el grupo, pero solo lectura para otros.
    - chmod -R o-rw \*: Elimina los permisos de lectura y escritura para otros (otros usuarios) de todos los archivos y directorios en el directorio actual y sus subdirectorios.

chown y chgrp (Change Owner y Change Group):

* Estos comandos se utilizan para cambiar el propietario o el grupo de un archivo o directorio. Por ejemplo:
  + - chown nuevo\_usuario archivo: Cambia el propietario del archivo al "nuevo\_usuario".
    - chgrp nuevo\_grupo archivo: Cambia el grupo del archivo al "nuevo\_grupo".

---------------------------------------------------------------------------------------------------------------

Señales

Los programas POSIX siguen un estándar para actuar cuando reciben ciertas señales. Sirven para interactuar con los procesos y para que ellos interactúen entre sí de ser necesario.

man 7 signal:

* Este comando muestra la página de manual (manual page) sobre las señales en el sistema. Proporciona información detallada sobre las señales y cómo se utilizan en los sistemas Unix/Linux. Puedes encontrar información sobre las diferentes señales y sus números de señal asociados, así como las situaciones en las que se envían y cómo se pueden manejar.

Para ver la página de manual sobre señales, ejecuta: man 7 signal Esto abrirá la página de manual en la que puedes navegar y obtener información detallada sobre las señales.

kill -l:

* Este comando muestra una lista de las señales disponibles en el sistema, junto con sus números de señal correspondientes. Es útil cuando necesitas conocer los nombres y números de señal de las señales disponibles en tu sistema. La salida puede ser útil para comprender cómo se identifican las señales en los comandos que las utilizan, como el comando kill.

Para ver la lista de señales y sus números de señal, ejecuta: kill -l Esto mostrará una lista de señales en la terminal.

-----------------------------------------------------------------------------------------------------------------

sleep 500 &

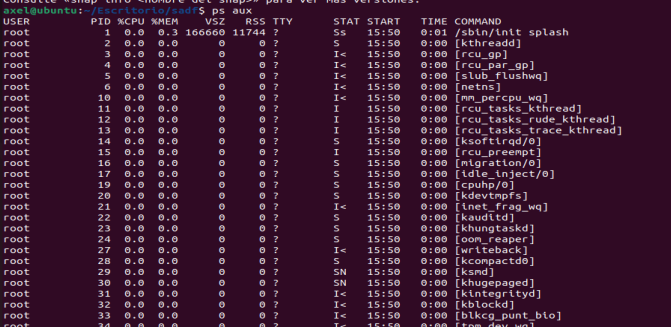
* El comando "sleep 500 &" se usa para hacer que la computadora espere durante 500 segundos (aproximadamente 8 minutos y 20 segundos) y ejecutarlo en segundo plano. Esto significa que la computadora esperará durante el tiempo especificado y luego continuará con otras tareas sin bloquearse.

Comandos Fundamentales – 5

Manipulación de procesos:

ps (Process Status):

* El comando ps se utiliza para mostrar información sobre los procesos en ejecución en el sistema. Algunas opciones comunes incluyen:
  + - ps aux: Muestra una lista detallada de todos los procesos en ejecución en el sistema, incluyendo información sobre el usuario que los inició, el uso de CPU, la memoria, etc.



En este ejemplo:

* La primera columna (USUARIO) muestra el nombre de usuario dueño del proceso.
* La segunda columna (PID) lista el ID del proceso.
* La tercera columna (%CPU) muestra el porcentaje de uso de CPU.
* La cuarta columna (%MEM) muestra el porcentaje de uso de memoria.
* La quinta columna (VSZ) representa el tamaño de memoria virtual en kilobytes.
* La sexta columna (RSS) indica el tamaño del conjunto residente (memoria en RAM) en kilobytes.
* La séptima columna (TTY) muestra el tipo de terminal o "?" si no está asociado a una terminal.
* La octava columna (ESTADO) representa el estado del proceso.
* La novena columna (COMIENZO) muestra la hora de inicio del proceso.
* La décima columna (TIEMPO) muestra el tiempo acumulado de CPU utilizado por el proceso.
* La undécima columna (COMANDO) lista el nombre del comando o programa asociado al proceso.

Esta salida proporciona una visión general de los procesos actualmente en ejecución en el sistema, con cada fila representando un proceso diferente.

* ps fax: Muestra los procesos en forma de un árbol jerárquico que refleja las relaciones de parentesco entre ellos.

top:

* El comando top muestra una lista en tiempo real de los procesos en ejecución y ofrece información actualizada sobre el uso de la CPU, la memoria y otros recursos del sistema. Es útil para monitorear el rendimiento del sistema en tiempo real.

Esto iniciará la aplicación top y mostrará una pantalla en tiempo real que se actualiza periódicamente. La información que se muestra en la parte superior de la pantalla incluye:

* La hora actual.
* El tiempo que el sistema ha estado encendido.
* La cantidad de usuarios conectados.
* La carga del sistema (carga promedio en los últimos 1, 5 y 15 minutos).
* La parte principal de la pantalla muestra una lista de procesos en ejecución, ordenados por diferentes métricas, como uso de CPU, memoria y más. Puedes utilizar las teclas de función o las teclas de flecha para interactuar con top y cambiar las opciones de visualización.

Algunos comandos comunes que puedes utilizar mientras top está en ejecución incluyen:

* q: Salir de top.
* k: Matar un proceso. Debes ingresar el número de proceso y presionar Enter.
* r: Cambiar la prioridad de un proceso.
* Space: Actualizar la pantalla.
* 1: Mostrar la vista de resumen de CPU.
* m: Mostrar la vista de resumen de memoria.
* t: Mostrar la vista de resumen de procesos.

top es una herramienta muy versátil y te proporciona información detallada sobre la actividad del sistema en tiempo real. Es particularmente útil cuando necesitas identificar procesos que consumen recursos de manera excesiva o cuando deseas monitorear el rendimiento de tu sistema en un entorno de producción. Puedes salir de top en cualquier momento presionando la tecla "q"

kill:

* El comando kill se utiliza para enviar señales a los procesos y puede utilizarse para detener o controlar su comportamiento. Puedes especificar una señal y el ID del proceso al que deseas enviar la señal.
  + kill -INT 312: Envía la señal INT (interrupción) al proceso con ID 312.
  + kill -USR1 2093: Envía la señal USR1 al proceso con ID 2093.
  + killall -TERM proceso: Envía la señal TERM (terminación) a todos los procesos con el nombre "proceso".
  + kill -l: Lista las señales posibles que se pueden utilizar con el comando kill. Puedes consultar las descripciones de las señales en las páginas del manual (man 7 signal).

sudo:

* El comando sudo se utiliza para ejecutar comandos con privilegios de superusuario o root. Puedes utilizar sudo antes de un comando para ejecutar ese comando con los permisos necesarios para realizar tareas administrativas en el sistema.
* sudo!!: Este comando ejecuta el comando anterior con privilegios de superusuario sin tener que volver a escribirlo. Es útil cuando te das cuenta de que necesitas permisos elevados después de haber ejecutado un comando.

El comando apt-get install :

* se utiliza en sistemas basados en Debian (como Ubuntu) para instalar paquetes de software desde los repositorios de software de la distribución. Este comando es parte del sistema de gestión de paquetes APT (Advanced Package Tool) y se utiliza para descargar, instalar y gestionar software de manera eficiente en sistemas Linux.

La sintaxis básica del comando es la siguiente: sudo apt-get install nombre\_del\_paquete

Donde:

* sudo se utiliza para ejecutar el comando con privilegios de superusuario, ya que la instalación de software generalmente requiere permisos elevados.
* apt-get es el comando para gestionar paquetes con APT.
* install es la acción que le indica a APT que instale el paquete.
* nombre\_del\_paquete es el nombre del paquete que deseas instalar.

Ejemplo

instalar varios programas

sudo apt-get install build-essential automake cmake

sí deseas instalar el paquete "curl" en Ubuntu, puedes hacerlo de la siguiente manera: sudo apt-get install curl

* El sistema buscará el paquete "curl" en los repositorios disponibles, lo descargará e instalará en tu sistema. Puedes utilizar este comando para instalar una amplia variedad de software en sistemas basados en Debian, lo que facilita la gestión de aplicaciones y herramientas en tu sistema.

Desistalar un programa

[Para desinstalar un programa en Linux, puedes seguir estos pasos1](https://computerhoy.com/noticias/software/linux-desinstalar-programa-terminal-79359)[2](https://www.genbeta.com/linux/como-desinstalar-aplicaciones-gnu-linux)[3](https://www.donprofe.com/como-desinstalar-un-programa-en-linux-desde-la-terminal):

1. **Abre la terminal**: Puedes hacerlo buscando “terminal” en el menú de tu sistema operativo o utilizando el atajo de teclado Ctrl+Alt+T en la mayoría de las distribuciones de Linux.
2. [**Lista tus programas instalados**: Escribe dpkg --list en la terminal y presiona Enter](about:blank)[3](https://www.donprofe.com/como-desinstalar-un-programa-en-linux-desde-la-terminal). Esto te mostrará una lista de todos los programas instalados actualmente en tu sistema.
3. [**Busca el programa que deseas desinstalar**: En la lista que apareció en el paso anterior, busca el nombre exacto del programa que deseas desinstalar3](https://www.donprofe.com/como-desinstalar-un-programa-en-linux-desde-la-terminal).
4. [**Desinstala el programa**: Escribe sudo apt-get --purge remove nombre-del-programa en la terminal](about:blank)[1](https://computerhoy.com/noticias/software/linux-desinstalar-programa-terminal-79359). Reemplaza “nombre-del-programa” con el nombre exacto del programa que encontraste en el paso 3.
5. **Confirma la desinstalación**: Se te pedirá que confirmes la desinstalación. [Para hacerlo, escribe “S” y presiona Enter1](https://computerhoy.com/noticias/software/linux-desinstalar-programa-terminal-79359).

sudo snap remove nombre-del-paquete

-----------------------------------------------------------------------------------------------------

Comandos Fundamentales – 6

cut:

* El comando cut se utiliza para mostrar columnas o campos específicos de un archivo de texto delimitado. Puedes especificar el delimitador con la opción -d y luego utilizar la opción -f para seleccionar la columna que deseas mostrar. Por ejemplo, para mostrar la quinta columna de un archivo "/etc/passwd" (un archivo de contraseñas y usuarios del sistema), puedes hacerlo de la siguiente manera: cut -d: -f5 /etc/passwd

tr:

* El comando tr se utiliza para realizar transformaciones de caracteres en un flujo de entrada. Puedes usarlo para cambiar mayúsculas a minúsculas o realizar otras operaciones de sustitución de caracteres. El ejemplo que proporcionaste cambia todas las letras mayúsculas a minúsculas en el contenido de "/etc/passwd".

adduser:

* El comando adduser se utiliza para agregar usuarios al sistema. Es una herramienta amigable que maneja aspectos como la creación de directorios de inicio y la asignación de grupos por defecto. En algunas distribuciones de Linux, como Ubuntu, adduser es una interfaz más fácil de usar que useradd.

addgroup:

* El comando addgroup se utiliza para agregar grupos al sistema. Puedes usarlo para crear grupos específicos para organizar a los usuarios. Sin embargo, ten en cuenta que la sintaxis y los comandos exactos pueden variar según la distribución de Linux.

usermod:

* El comando usermod se utiliza para modificar las propiedades de un usuario existente en el sistema. Puedes usarlo para cambiar el nombre de usuario, grupo principal, directorio de inicio y otras propiedades de un usuario.

passwd:

* El comando passwd se utiliza para modificar la contraseña de un usuario. Puedes usarlo para cambiar tu propia contraseña o, si tienes privilegios de superusuario, para cambiar la contraseña de otros usuarios.

Ejemplos

* **cut**: Mostrar el nombre de usuario (primer campo) y el directorio de inicio (septimo campo) de **/etc/passwd**: cut -d: -f 1,7 /etc/passwd
* **tr**: Convertir un archivo de texto a minúsculas: tr '[:upper:]' '[:lower:]' < archivo.txt
* **adduser** (o **useradd**): Agregar un nuevo usuario al sistema (reemplaza "nuevo\_usuario" con el nombre que desees): adduser nuevo\_usuario
* **addgroup** (o **groupadd**): Agregar un nuevo grupo al sistema (reemplaza "nuevo\_grupo" con el nombre que desees): addgroup nuevo\_grupo
* **usermod**: Cambiar el nombre completo de un usuario (reemplaza "usuario" y "Nuevo Nombre" con los valores adecuados): usermod -c "Nuevo Nombre" usuario
* **passwd**: Cambiar la contraseña de un usuario (reemplaza "usuario" con el nombre de usuario correspondiente): passwd usuario Principio del formulario

----------------------------------------------------------------------------------------------------

Comandos Fundamentales – 7

El comando find en sistemas Unix/Linux se utiliza para buscar archivos y directorios que coincidan con ciertos criterios de búsqueda en un sistema de archivos. Es una herramienta poderosa para buscar archivos basados en nombres, ubicaciones, fechas de modificación y otros atributos. Aquí hay algunos ejemplos de cómo usar find:

Búsqueda por nombre de archivo:

* Para buscar un archivo específico por su nombre en todo el sistema de archivos, puedes usar el siguiente comando. Por ejemplo, para encontrar un archivo llamado "cosa.txt": find / -name "cosa.txt"

Búsqueda por fecha de modificación:

* Para buscar archivos modificados en los últimos n días (en este caso, 1 día), puedes usar -mtime. Por ejemplo, para encontrar archivos modificados en las últimas 24 horas: find / -mtime -1

Búsqueda por tamaño y acción:

* Puedes buscar archivos que cumplan con ciertos criterios de tamaño y realizar una acción en ellos. Por ejemplo, el siguiente comando busca archivos en el directorio actual que tengan más de 10 días de antigüedad y que sean más grandes que 50,000 bytes, y luego los elimina: find . -mtime +10 -size +50000c -exec rm {} \;

En este caso, find encuentra los archivos que coinciden con los criterios y ejecuta el comando rm en cada uno de ellos.

locate:

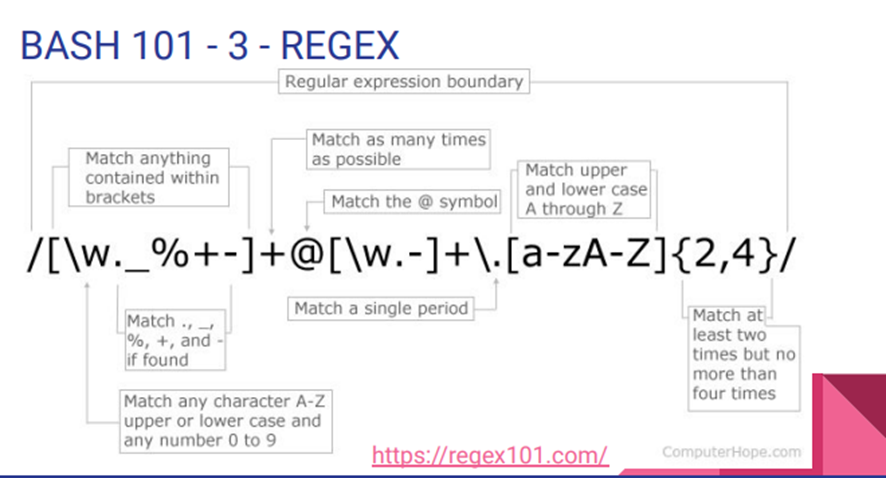
* El comando locate es otra herramienta para buscar archivos, pero a diferencia de find, utiliza una base de datos precompilada para realizar búsquedas rápidas. Es más rápido que find, pero puede no ser tan preciso en tiempo real, ya que su base de datos se actualiza periódicamente. Para usar locate, simplemente proporciona el nombre del archivo que deseas buscar: locate nombre\_de\_archivo

Ten en cuenta que debes ejecutar sudo updatedb para actualizar la base de datos de locate si no encuentra un archivo que esperas encontrar.

find y locate son herramientas útiles para buscar archivos y directorios en sistemas Unix/Linux, y la elección entre ellos depende de tus necesidades específicas y de la precisión en tiempo real que requieras en la búsqueda.

------------------------------------------------------------------------------------------------------------------

[regex101: build, test, and debug regex](https://regex101.com/)



1. Sintaxis básica de una expresión regular:

* Las expresiones regulares se escriben entre barras inclinadas (/) o comillas (" o '). Por ejemplo:
  + /patrón/
  + 'patrón'

2. Coincidencia de caracteres literales:

* Puedes usar caracteres literales para buscar una coincidencia exacta en un texto. Por ejemplo, /hello/ buscará la palabra "hello" en un texto.

3. Meta caracteres:

* Los meta caracteres son caracteres especiales que tienen significados especiales en expresiones regulares. Algunos de los meta caracteres comunes son:
  + .: Coincide con cualquier carácter, excepto el salto de línea.
  + \*: Coincide con cero o más repeticiones del carácter o patrón anterior.
  + +: Coincide con una o más repeticiones del carácter o patrón anterior.
  + ?: Coincide con cero o una repetición del carácter o patrón anterior.
  + []: Coincide con un solo carácter que esté dentro de los corchetes. Por ejemplo, [aeiou] coincidirá con cualquier vocal.
  + [^]: Coincide con un solo carácter que NO esté dentro de los corchetes. Por ejemplo, [^0-9] coincidirá con cualquier carácter que NO sea un número.

4. Secuencias de escape:

* Puedes usar secuencias de escape para buscar caracteres que son meta caracteres. Por ejemplo, \., \\*, y \? coincidirán con los caracteres . (punto), \* (asterisco) y ? (signo de interrogación) de manera literal.

5. Cuantificadores:

* Los cuantificadores especifican cuántas veces debe aparecer un carácter o patrón en una coincidencia. Algunos ejemplos de cuantificadores son {n}, {n,}, {n,m}. Por ejemplo, /a{3}/ coincidirá con "aaa", pero no con "aa".

6. Anclajes:

* Los anclajes se utilizan para especificar dónde debe aparecer un patrón en el texto. Algunos ejemplos de anclajes son ^ (inicio de línea) y $ (fin de línea). Por ejemplo, /^hello/ coincidirá con "hello" solo si está al principio de una línea.

7. Operadores lógicos:

* Puedes combinar expresiones regulares con operadores lógicos como | (OR) y () (paréntesis) para crear patrones más complejos. Por ejemplo, /patrón1|patrón2/ coincidirá con "patrón1" o "patrón2".

8. Aplicación de expresiones regulares en BASH:

* En BASH, puedes usar expresiones regulares en comandos como grep, sed y awk para buscar, reemplazar y manipular texto de acuerdo con un patrón dado.

Aquí hay un ejemplo simple de cómo usar grep con una expresión regular en BASH para buscar líneas que contengan la palabra "hello" en un archivo de texto: grep 'hello' archivo.txt

Este es solo un vistazo básico a las expresiones regulares en BASH. Puedes profundizar más en este tema para realizar búsquedas y manipulaciones más avanzadas de texto en BASH.

------------------------------------------------------------------------------------------------------------------

BASH 101 – 3

? y \* son comodines utilizados en expresiones regulares y patrones de búsqueda en BASH y otros lenguajes para representar caracteres o secuencias de caracteres desconocidos o variables en un texto. Aquí tienes una explicación más detallada de cómo funcionan:

1. ? (Signo de interrogación):
   * El símbolo ? se utiliza como comodín para representar un solo carácter en una expresión regular. Por ejemplo: a?e coincidirá con "ane", "abe", "ace", etc., donde el ? representa un solo carácter en esa posición.
2. \* (Asterisco):
   * El asterisco \* se utiliza como comodín para representar cero o más caracteres en una expresión regular. Por ejemplo: ar\* coincidirá con "ar", "are", "arr", "argentina", "arbol", etc., donde el \* representa cero o más caracteres después de "ar".
3. Ejemplos adicionales:

* \*r\* coincidirá con cualquier cadena que contenga al menos una "r" en cualquier posición.
* coincidirá con cualquier cadena, ya que el \* representa cero o más caracteres.
* ?d\* coincidirá con palabras que comienzan con un carácter seguido de "d" y luego cero o más caracteres adicionales, como "admin", "administracion", "bd", "edificio", etc.
* un?am coincidirá con palabras que comienzan con "un", luego cualquier carácter, seguido de "am", como "unsam", "unlam", "unxam", "un5am", etc.

Claro, aquí tienes algunos ejemplos prácticos de cómo utilizar los comodines ? y \* en expresiones regulares en BASH:

Usando ? para un solo carácter:

* Buscar palabras que contengan una vocal seguida de una "r" grep '[aeiou]r' archivo.txt

Esto buscará palabras que tengan una vocal seguida de la letra "r" en un archivo de texto. Usando \* para cero o más caracteres:

* Buscar palabras que comiencen con "auto" y puedan tener caracteres adicionales grep 'auto.\*' archivo.txt

Esto buscará palabras que comiencen con "auto" y puedan tener cualquier cantidad de caracteres adicionales después de "auto" en un archivo de texto. Usando ambos comodines juntos:

* Buscar palabras que comiencen con una vocal, tengan cualquier cantidad de caracteres en el medio y terminen con la letra "o" grep '[aeiou]\*o' archivo.txt

Esto buscará palabras que comiencen con una vocal, tengan cualquier cantidad de caracteres en el medio y terminen con la letra "o" en un archivo de texto.

Usando comodines en un patrón más complejo:

* Buscar números de teléfono en formato XXX-XXXX o XXX-XXXXX grep '[0-9]\{3\}-[0-9]\{4,5\}' archivo.txt

Esto buscará números de teléfono en formato "XXX-XXXX" o "XXX-XXXXX" en un archivo de texto, donde \{3\} indica exactamente 3 dígitos y \{4,5\} indica de 4 a 5 dígitos.

--------------------------------------------------------------------------------------------------------------------

BASH 101 – 4

los corchetes [ ] y el signo de exclamación ! dentro de expresiones regulares en BASH se utilizan para realizar selecciones más específicas de caracteres. Aquí hay algunos ejemplos adicionales para ilustrar su uso:

1. [abc]\*: Esto coincidirá con cualquier cadena que comience con uno de los caracteres "a," "b," o "c," seguido de cero o más caracteres adicionales.
2. [!abc]\*: Esto coincidirá con cualquier cadena que NO comience con ninguno de los caracteres "a," "b," o "c."
3. [A-Z]\*: Esto coincidirá con cualquier cadena que comience con una letra mayúscula seguida de cero o más caracteres adicionales.
4. funcion.[ch]: Esto coincidirá con "funcion.c" o "funcion.h," ya que [ch] representa una selección de uno de los caracteres "c" o "h."
5. [Uu]nsam: Esto coincidirá con "Unsam" o "unsam," ya que [Uu] representa una selección de uno de los caracteres "U" o "u."

Estos corchetes [ ] son útiles para especificar conjuntos de caracteres permitidos o excluidos en patrones de búsqueda, lo que permite realizar coincidencias más precisas en texto. También puedes utilizar rangos de caracteres dentro de los corchetes, como [A-Z] para representar cualquier letra mayúscula en el alfabeto

Ejemplo: ls /bin/[abc]\*

--------------------------------------------------------------------------------------------------------

BASH 101 – 5

Las llaves { } se pueden utilizar para generar listas de opciones o alternativas en patrones de búsqueda o para definir alias de comandos. Aquí tienes algunos ejemplos adicionales de cómo funcionan las llaves { } y cómo se pueden usar los alias:

Llaves para generar listas de opciones:

* Puedes usar las llaves { } para generar una lista de opciones o alternativas en un patrón de búsqueda. Por ejemplo: Fichero.{cpp,h} coincidirá con "Fichero.cpp" o "Fichero.h".

Llaves anidadas:

* También puedes anidar las llaves dentro de otras llaves para generar combinaciones más complejas. Por ejemplo: b{ue{no,na},oca}s coincidirá con "buenos", "buenas", y "bocas".

Alias de comandos:

* Puedes crear alias de comandos personalizados utilizando la palabra clave alias. Esto te permite definir versiones abreviadas o personalizadas de comandos comunes para simplificar tu flujo de trabajo. Por ejemplo: alias ls='ls -l' crea un alias para el comando ls, de modo que cuando escribas ls, en realidad estás ejecutando ls -l.

Alias con opciones personalizadas:

* También puedes usar alias para agregar opciones personalizadas a comandos. Por ejemplo: alias rm='rm --interactive --verbose' agrega las opciones --interactive y --verbose al comando rm, de modo que cuando ejecutes rm, se ejecutará con esas opciones.

Estos son ejemplos de cómo las llaves { } y los alias pueden mejorar la flexibilidad y la personalización de tu experiencia en la línea de comandos en sistemas Unix/Linux. Los alias son especialmente útiles para crear atajos para comandos largos o para agregar opciones predeterminadas a comandos que usas con frecuencia.

----------------------------------------------------------------------------------------------------------------

BASH 101 – 6

Variables de entorno (Environment variables)

Las variables de entorno en BASH son elementos fundamentales para configurar y personalizar el entorno de tu sesión o sistema. Aquí hay algunos aspectos clave sobre las variables de entorno:

1. Contiene texto: Las variables de entorno almacenan información en forma de texto, como rutas de directorio, configuraciones de aplicación, nombres de usuario, etc.
2. Creación: Para crear una variable de entorno, puedes usar el formato VAR=mi\_variable, donde "VAR" es el nombre de la variable y "mi\_variable" es su valor. No deben haber espacios alrededor del signo igual (=).
3. Uso: Para acceder al valor de una variable de entorno, utiliza el signo de dólar seguido del nombre de la variable, como $VAR. Por ejemplo, echo $VAR mostrará el valor de la variable "VAR".
4. Locales a un proceso a menos que se exporten: Por defecto, las variables de entorno son locales a un proceso y no están disponibles para otros procesos. Esto significa que si defines una variable en un script o terminal, no estará disponible en otros scripts o terminales. Para hacer una variable de entorno global, puedes usar el comando export, por ejemplo, export VAR.
5. Para BASH: Si deseas que una variable de entorno esté disponible para todos los subprocesos de tu sesión de BASH, debes usar export al definirla, como en export VAR=mi\_variable.

Las variables de entorno son útiles para configurar tu entorno de trabajo, establecer preferencias de usuario y almacenar información que pueda ser utilizada por varios programas y scripts en tu sistema.

-------------------------------------------------------------------------------------------------------------------

BASH 101 - 6

Variables de entorno particulares

1. $USER: Esta variable almacena el nombre del usuario actual que ha iniciado sesión en el sistema.
2. $HOME: Contiene la ruta al directorio principal del usuario actual, es decir, el directorio en el que se encuentra cuando inicia sesión.
3. $SHELL: Indica la ruta al intérprete de comandos (shell) que está utilizando el usuario. Por ejemplo, "/bin/bash" para BASH.
4. $PWD: Almacena la ruta completa del directorio de trabajo actual, es decir, el directorio en el que te encuentras actualmente.
5. $OLDPWD: Mantiene la ruta completa del directorio de trabajo anterior, es decir, el directorio en el que estabas antes de cambiar al directorio actual.
6. $PATH: Esta variable almacena una lista de rutas (directorios) separadas por dos puntos (:). Cada ruta en esta lista es el lugar donde el sistema buscará ejecutables cuando ingreses un comando en el terminal. Es fundamental para encontrar programas y comandos en tu sistema.
7. $?: Esta variable almacena el código de salida del último programa o comando ejecutado. Un valor de 0 generalmente indica éxito, mientras que otros valores pueden indicar errores u otros estados.

Estas variables de entorno son esenciales para la configuración y el funcionamiento de tu entorno de shell en sistemas Unix y Linux. Puedes utilizarlas en scripts y comandos para realizar tareas específicas o para personalizar tu entorno de trabajo.

------------------------------------------------------------------------------------------------------------------

BASH 101 - 7

Prompt

http://ezprompt.net/

El prompt (PS1) en una terminal de Unix/Linux se refiere al formato del símbolo o texto que se muestra antes de que ingreses un comando. Puedes personalizar tu prompt utilizando diversas secuencias de escape para mostrar información útil como la fecha, hora, nombre de la máquina, nombre de usuario y directorio actual. Aquí está el ejemplo que proporcionaste: PS1='\d-\t:\h|\w $ '

Desglose de este prompt personalizado:

* \d: Se sustituye por la fecha actual.
* \t: Se sustituye por la hora actual.
* \h: Se sustituye por el nombre de la máquina (host).
* \w: Se sustituye por el directorio de trabajo actual.
* $: Indica el final del prompt y el lugar donde puedes ingresar comandos.

Entonces, cuando ejecutas este PS1 personalizado, obtienes un prompt similar a este: mon. febr. 04-10:25:09:tpi12.unsam.edu.ar|~ $

* "mon. febr. 04" es la fecha.
* "10:25:09" es la hora.
* "tpi12.unsam.edu.ar" es el nombre de la máquina.
* "~" es el directorio actual.
* "$" es el símbolo que indica que puedes ingresar comandos.

Esta personalización del prompt te permite tener información útil a la vista mientras trabajas en tu terminal. Puedes ajustar el formato del PS1 según tus preferencias para que se adapte mejor a tus necesidades.

--------------------------------------------------------------------------------------------------------

BASH 101 – 8

El archivo "~/.bashrc" es un archivo de configuración en sistemas Unix y Linux que se ejecuta automáticamente cada vez que se inicia una nueva sesión de terminal para un usuario. Es utilizado para personalizar el entorno de la shell Bash para ese usuario específico.

* Guardar la configuración.
* Corre cada vez que llamamos a la terminal
* Buen lugar donde guardar las variables personalizadas.

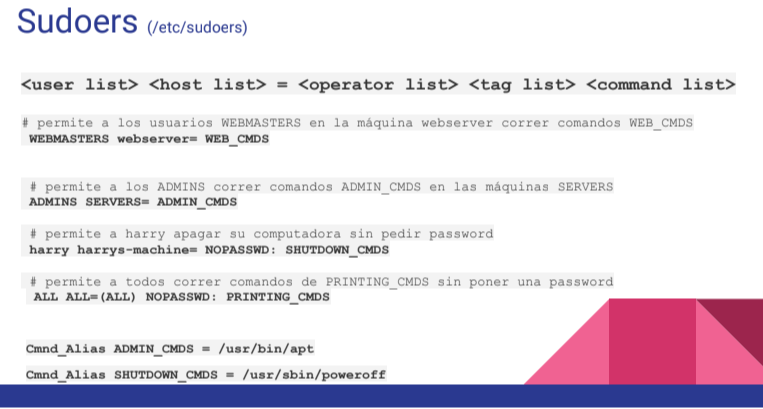
Algunas de las cosas comunes que puedes hacer en "~/.bashrc" incluyen:

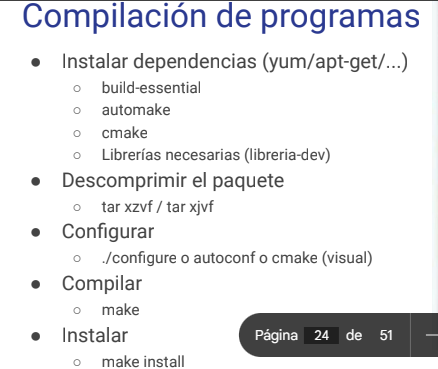
1. Definir alias de comandos: Puedes crear alias cortos para comandos largos o personalizar la forma en que se ejecutan ciertos comandos.
2. Configurar el prompt (PS1): Puedes personalizar el aspecto del prompt de tu terminal, como se mencionó en la respuesta anterior.
3. Definir variables de entorno: Puedes establecer variables de entorno específicas para tu sesión de Bash, que serán válidas cada vez que inicies una nueva sesión.
4. Configurar rutas (PATH): Puedes agregar directorios al PATH, lo que te permite ejecutar programas desde esos directorios sin especificar la ruta completa.
5. Personalizar colores y formato de salida: Puedes cambiar los colores y la apariencia de los mensajes y resultados de comandos en la terminal.
6. Definir funciones de Bash personalizadas: Puedes crear funciones de Bash para realizar tareas específicas o simplificar secuencias de comandos.
7. Cargar archivos adicionales: Puedes usar "~/.bashrc" para cargar otros archivos de configuración si es necesario.

Para editar el archivo "~/.bashrc", puedes utilizar un editor de texto como "nano" o "vim" desde la línea de comandos. Asegúrate de guardar los cambios después de realizar modificaciones en este archivo para que surtan efecto en futuras sesiones de terminal.

Recuerda que cualquier cambio que realices en "~/.bashrc" solo afectará a tu sesión de Bash actual y a las futuras sesiones de Bash que inicies como ese usuario.

--------------------------------------------------------------------------------------------





--------------------------------------------------------------------------------------------------------

Procesos y Jobs

* Jobs es una nomenclatura paralela a proceso
* Bash administra una lista de procesos en cola
* comando & : crea un job
* jobs : ver los jobs actuales
* Ctrl-z : para el proceso actual
* fg : pone un proceso parado en foreground
* bg : pone un proceso parado en background

Notas :

* Ctrl-c : interrumpe definitivamente el proceso corriendo
* Ctrl-d : interrumpe definitivamente la entrada por teclado
* nohup : para crear procesos que no estén atados al shell actual

Los "jobs" en el contexto de la línea de comandos de Bash se refieren a la gestión de procesos en segundo plano. Aquí tienes una descripción de los conceptos que mencionaste:

* Comando &: Al agregar "&" al final de un comando, se ejecutará en segundo plano como un job. Esto significa que el proceso se ejecutará en segundo plano y podrás seguir utilizando tu terminal para ingresar otros comandos.
* jobs: El comando "jobs" se utiliza para ver una lista de los jobs actuales en ejecución en tu sesión de Bash. Te muestra el número de job, el estado (en segundo plano o en primer plano), y el comando asociado.
* Ctrl-z: Al presionar "Ctrl-z" mientras un proceso se está ejecutando en primer plano, se detendrá ese proceso y se colocará en segundo plano como un job detenido.
* fg: El comando "fg" se utiliza para traer un job detenido al primer plano y reanudar su ejecución en el primer plano. Puedes especificar el número del job o el identificador para indicar cuál job traer al primer plano si hay varios.
* bg: El comando "bg" se utiliza para reanudar la ejecución de un job detenido en segundo plano. Al igual que con "fg", puedes especificar el número del job o el identificador.
* Ctrl-c: "Ctrl-c" se utiliza para interrumpir definitivamente el proceso en primer plano que se está ejecutando. Esto enviará una señal SIGINT al proceso y lo finalizará.
* Ctrl-d: "Ctrl-d" se utiliza para indicar el final de la entrada estándar (EOF) en una terminal. En un contexto interactivo, esto generalmente significa que estás cerrando la terminal.
* nohup: El comando "nohup" se utiliza para ejecutar un proceso que no esté vinculado al shell actual. Esto significa que el proceso seguirá ejecutándose incluso si cierras la terminal o la sesión de shell. Es útil para procesos que deben continuar ejecutándose después de que te desconectes.

La gestión de jobs en Bash es útil para realizar tareas en segundo plano, detener o reanudar procesos y administrar la ejecución de comandos en tu sesión de shell.

--------------------------------------------------------------------------------------------------------------------

El comando que proporcionaste, disown -h %1, se utiliza en la línea de comandos de Unix o Linux para desvincular un trabajo (job) del terminal actual y evitar que sea enviado a primer plano cuando se cierra la terminal o se inicia una nueva sesión. Aquí hay una explicación detallada de lo que hace cada parte del comando:

* disown: Este comando se utiliza para quitar trabajos (jobs) del shell, lo que significa que se elimina la relación entre el trabajo y la terminal.
* -h: La opción -h se usa para marcar un trabajo para que no se envíe a primer plano cuando se cierra la terminal. Esto permite que el trabajo continúe ejecutándose en segundo plano incluso después de cerrar la terminal.
* %1: Esto se refiere al primer trabajo en segundo plano. Los trabajos en segundo plano se numeran automáticamente, y %1 se refiere al primer trabajo en segundo plano.

Entonces, cuando ejecutas disown -h %1, estás diciéndole al sistema que tome el primer trabajo en segundo plano (identificado como %1) y lo desvincule del terminal actual, permitiendo que el trabajo continúe ejecutándose incluso si cierras la terminal.

ejemplo de cómo usar dpkg -l para listar todos los paquetes instalados: dpkg -l

Esta lista contendrá información como el estado del paquete (instalado, desinstalado, etc.), el nombre del paquete, la versión, la arquitectura y una breve descripción.

* Ten en cuenta que el resultado puede ser extenso, por lo que es común usar herramientas como grep para buscar paquetes específicos o redirigir la salida a un archivo para su posterior referencia. Por ejemplo: dpkg -l | grep nombre\_del\_paquete

Esto buscará el paquete con el nombre especificado en la lista generada por dpkg -l.

----------------------------------------------------------------------------------------------------------------

realizar operaciones como búsqueda y coincidencia de archivos o texto. Aquí hay una breve explicación de cada uno de tus ejemplos:

1. Fichero.{cpp,h}: Esta expresión regular coincide con "Fichero.cpp" y "Fichero.h", lo que significa que se usaría para buscar o hacer referencia a archivos con nombres "Fichero.cpp" y "Fichero.h". Los corchetes y las comas indican opciones.
2. b{ue{no,na},oca}s: Esto se expande a "buenos", "buenas" y "bocas". Las llaves y comas aquí permiten que se generen múltiples combinaciones de palabras.
3. c{inta,ancion,andombe}: Esto se expande a "cinta", "cancion" y "candombe". Nuevamente, las llaves y comas se utilizan para generar múltiples opciones de palabras.

Estas expresiones regulares son útiles para buscar y seleccionar patrones específicos de texto o archivos cuando se trabaja en la línea de comandos o en scripts

ejemplos de cómo se pueden usar las expresiones regulares del ejemplo 2:

1. Para buscar archivos que contienen palabras como "buenos", "buenas" o "bocas" en sus nombres: ls \*b{ue{no,na},oca}s\*

Esto listará archivos que tengan cualquiera de esas palabras en su nombre.

1. Para buscar líneas en un archivo de texto que contienen palabras como "buenos", "buenas" o "bocas": grep 'b\(ue\(no\|na\)\|ocas\)' archivo.txt

Esto buscará y mostrará las líneas que contienen alguna de las palabras especificadas en el archivo de texto.

1. Para hacer coincidir palabras en un texto usando herramientas como sed: echo "Me gustan los buenos días" | sed 's/b\(ue\(no\|na\)\|ocas\)s/XXX/g'

* Esto reemplazará "buenos", "buenas" o "bocas" con "XXX" en la cadena de texto.
* Estos son solo algunos ejemplos de cómo las expresiones regulares pueden utilizarse en la línea de comandos para buscar y manipular texto o nombres de archivos. Las expresiones regulares son una herramienta poderosa para el procesamiento de texto y búsquedas avanzadas

-----------------------------------------------------------------------------------------------------------------

El comando "du" en Linux se utiliza para mostrar el uso de espacio en disco de archivos y directorios. Puedes usarlo de la siguiente manera: du [opciones] [archivo o directorio]

Algunas opciones comunes incluyen:

* -h: Muestra los resultados en un formato legible para humanos, utilizando unidades como KB, MB, GB, etc.
* -s: Muestra el resumen del uso de espacio solo para el directorio o archivo especificado, sin listar detalles de subdirectorios.
* -c: Muestra un total general al final de la lista.
* -m: Muestra los resultados en megabytes.
* -k: Muestra los resultados en kilobytes.

Por ejemplo, para ver el uso de espacio en disco de un directorio llamado "mi\_directorio", puedes ejecutar: du -h mi\_directorio

* Esto mostrará una lista de subdirectorios y archivos dentro de "mi\_directorio" junto con el uso de espacio en disco de cada uno en un formato legible para humanos.
* El comando "du" es útil para verificar qué archivos o directorios están ocupando más espacio en tu sistema de archivos y puede ser útil para la administración del espacio en disco.

-------------------------------------------------------------------------------------------------------------------

Strace

Debugger de procesos

Strace es una herramienta de depuración para procesos en sistemas operativos Linux. Permite rastrear las llamadas al sistema y las señales que recibe un proceso. Aquí te dejo una breve explicación de los comandos que mencionaste:

* ps -C firefox-bin: Este comando muestra los procesos que están ejecutando el programa firefox-bin.
* sudo strace -p 1725 -o firefox\_trace.txt: Este comando adjunta strace al proceso con el ID 1725 y guarda la salida en el archivo firefox\_trace.txt[1](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C0%7C08d8d23f-891e-4213-b88d-18f12046ad36%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22%E2%97%8F%20sudo%20str...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A26%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A4%2C%22highlightText%22%3A%22%E2%97%8F%20sudo%20strace%20-p%201725%20-o%20firefox_trace.txt%5Cr%5Cn%E2%97%8F%20strace%20-t%20-e%20open%20ls%20%2Fhome%22%2C%22snippets%22%3A%5B%5D%7D%7D).
* strace -t -e open ls /home: Este comando rastrea las llamadas al sistema ‘open’ realizadas por el comando ls /home y muestra la hora en que se realizó cada llamada.
* strace -c ls > /dev/null: Este comando rastrea las llamadas al sistema realizadas por el comando ls, cuenta la cantidad de veces que se realiza cada llamada y desecha la salida.

Principio del formulario

--------------------------------------------------------------------------------------------------------------------

Estás proporcionando información sobre dos potentes herramientas de procesamiento de texto en Unix/Linux: **SED** y **AWK**.

**SED** es un editor de flujo que permite realizar transformaciones básicas de texto en un flujo (es decir, una tubería) o desde un archivo. Algunos ejemplos que has mencionado son:

* sed 's/hola/hello/g' archivo : Este comando reemplaza todas las ocurrencias de “hola” por “hello” en el archivo.
* sed '/windows/d' archivo : Este comando borra todas las líneas que contienen la palabra “windows” en el archivo.
* sed '1,10d' archivo : Este comando borra las primeras 10 líneas del archivo.

**AWK** es otro lenguaje de programación diseñado para el procesamiento de datos basado en texto. AWK es particularmente útil cuando los datos están estructurados con campos y registros. Algunos ejemplos que has mencionado son:

* awk '/windows/{print $2, $1}' archivo : Este comando imprime el segundo y el primer campo de cada línea que contiene la palabra “windows” en el archivo.
* awk -f comandos archivos : Este comando ejecuta los comandos AWK guardados en el archivo “comandos” en los archivos especificados.
* ./script.awk < archivo : Este comando ejecuta un script AWK en el archivo especificado.

**Aquí tienes algunos ejemplos de cómo usar SED y AWK:**

**SED**

1. Reemplazar todas las ocurrencias de una cadena:
2. echo "hola mundo" | sed 's/hola/hello/g'

Este comando reemplaza “hola” por “hello”, produciendo la salida “hello mundo”.

1. Eliminar líneas que contienen una cadena específica:
2. echo -e "windows\nlinux\nmac" | sed '/windows/d'

Este comando elimina las líneas que contienen la palabra “windows”, por lo que la salida será:

linux

mac

1. Eliminar las primeras n líneas de un archivo (por ejemplo, las primeras 2 líneas):
2. echo -e "linea1\nlinea2\nlinea3" | sed '1,2d'

Este comando elimina las primeras 2 líneas, por lo que la salida será “linea3”.

**AWK**

1. Imprimir campos específicos (por ejemplo, el segundo y el primer campo):
2. echo -e "campo1 campo2\ncampo3 campo4" | awk '{print $2, $1}'

Este comando intercambia el orden de los campos, por lo que la salida será:

campo2 campo1

campo4 campo3

1. Ejecutar un script AWK en un archivo: Supongamos que tienes un script llamado script.awk con el siguiente contenido:
2. {print $1}

Puedes ejecutar este script en un archivo de la siguiente manera:

echo -e "campo1 campo2\ncampo3 campo4" | awk -f script.awk

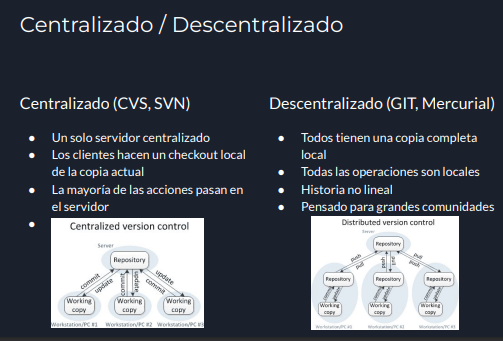
Este comando imprimirá el primer campo de cada línea[1](https://geekland.eu/uso-del-comando-awk-en-linux-y-unix-con-ejemplos/)[2](https://atareao.es/tutorial/terminal/filtros-awk-grep-sed-y-cut/)[3](https://programmerclick.com/article/1063790693/)[4](https://faqfreak.com/diferencia-entre-sed-y-awk/)[5](https://www.infor.uva.es/~mluisa/talf/docs/aula/A2.pdf).

--------------------------------------------------------------------------------------------------------------------

Pdf de versionado de código git

Versionado de código

* El versionado administra el cambio
* Permite a los equipos colaborar
* Permite la inspección
* Marca el ownership
* Marca la evolución de los cambios
* Permiten las ramificaciones
* Permite la integración continua



Snapshots, Commits

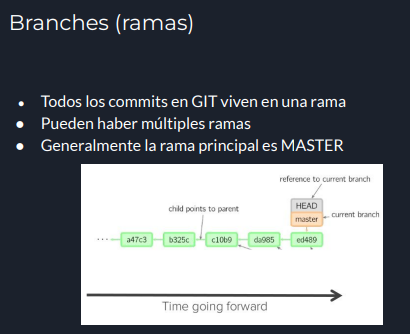
Snapshot: Es la manera de GIT para seguir los cambios. Guarda una imagen de todo tu código en un punto específico del tiempo. Se mantienen en el tiempo.

Commit: El acto de crear un snapshot. Contiene tres datos importantes:

* Información de como cambio cada archivo con referencia al anterior snapshot
* Un puntero al commit anterior
* Un código único identificatorio

Repositorios, cloning, pulling y pushing

* Colección de todos los archivos y las historias de los mismos.
* Puede estar en una máquina local o Remota (github, gitlab cloud, bitbucket…)
* El acto de copiar el código del servidor remoto al local se llama Cloning.
* El acto de bajar los commits que no existen en la copia local se llama pulling.
* El acto de subir los cambios locales al servidor remoto se llama pushing



Comandos Básicos

Estos son comandos básicos de **GIT**, una herramienta de control de versiones:

1. **GIT CLONE**: Este comando clona un repositorio remoto a un directorio local[1](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C0%7Cccf08d5b-e4cd-46aa-b696-5b612d4ca186%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22GIT%20CLONE%20...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A8%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F02%252520-%252520Versionado%252520de%252520codigo%252520%252520GIT.pdf%22%2C%22lineIndex%22%3A2%2C%22highlightText%22%3A%22GIT%20CLONE%20%3A%20clona%20un%20repositorio%20remoto%20a%20un%20path%20local.%22%2C%22snippets%22%3A%5B%5D%7D%7D). Es útil cuando quieres tener una copia del código en tu máquina local.
2. **GIT PULL**: Este comando actualiza tu repositorio local con cualquier cambio que se haya hecho en el repositorio remoto que aún no exista en tu copia local.
3. **GIT CHECKOUT**: Este comando se utiliza para cambiar a una rama diferente en tu repositorio local. También puedes usarlo para crear nuevas ramas.
4. **GIT ADD**: Este comando agrega archivos modificados al “área de preparación” en Git, que es un paso preliminar antes de confirmar los cambios en el repositorio.
5. **GIT COMMIT**: Este comando crea un nuevo “commit” o punto de control en tu repositorio con todos los cambios que hayas agregado con git add. Cada commit tiene un mensaje asociado que describe los cambios realizados.
6. **GIT PUSH**: Este comando sube los cambios que hayas hecho y confirmado en tu repositorio local al repositorio remoto.

Estos comandos son fundamentales para trabajar con Git y colaborar en proyectos de código.

**Aquí tienes algunos ejemplos de cómo usar los comandos de GIT que mencionaste:**

1. **GIT CLONE**: Este comando se utiliza para clonar (copiar) un repositorio remoto en tu máquina local. Por ejemplo:
2. git clone https://github.com/usuario/repositorio.git

Este comando clonará el repositorio remoto en un directorio local llamado “repositorio”.

1. **GIT PULL**: Este comando se utiliza para actualizar tu repositorio local con cualquier cambio que se haya hecho en el repositorio remoto. Por ejemplo:
2. git pull origin master

Este comando fusionará los cambios del branch “master” del repositorio remoto “origin” en tu branch actual.

1. **GIT CHECKOUT**: Este comando se utiliza para cambiar a una rama diferente en tu repositorio local. También puedes usarlo para crear nuevas ramas. Por ejemplo:
2. git checkout -b nueva\_rama

Este comando creará una nueva rama llamada “nueva\_rama” y cambiará a ella.

1. **GIT ADD**: Este comando agrega archivos modificados al “área de preparación” en Git, que es un paso preliminar antes de confirmar los cambios en el repositorio. Por ejemplo:
2. git add archivo\_modificado.txt

Este comando agregará el archivo “archivo\_modificado.txt” al área de preparación.

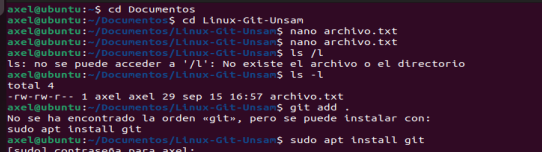
1. **GIT COMMIT**: Este comando crea un nuevo “commit” o punto de control en tu repositorio con todos los cambios que hayas agregado con git add. Cada commit tiene un mensaje asociado que describe los cambios realizados. Por ejemplo:
2. git commit -m "Descripción de los cambios realizados"

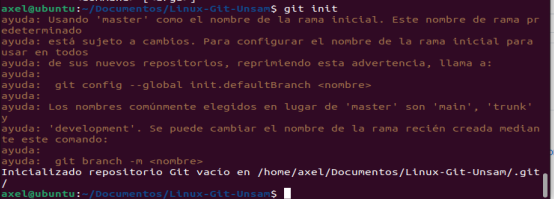
Este comando creará un nuevo commit con la descripción proporcionada.

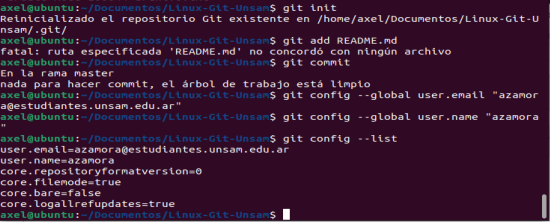
1. **GIT PUSH**: Este comando sube los cambios que hayas hecho y confirmado en tu repositorio local al repositorio remoto. Por ejemplo:
2. git push origin master

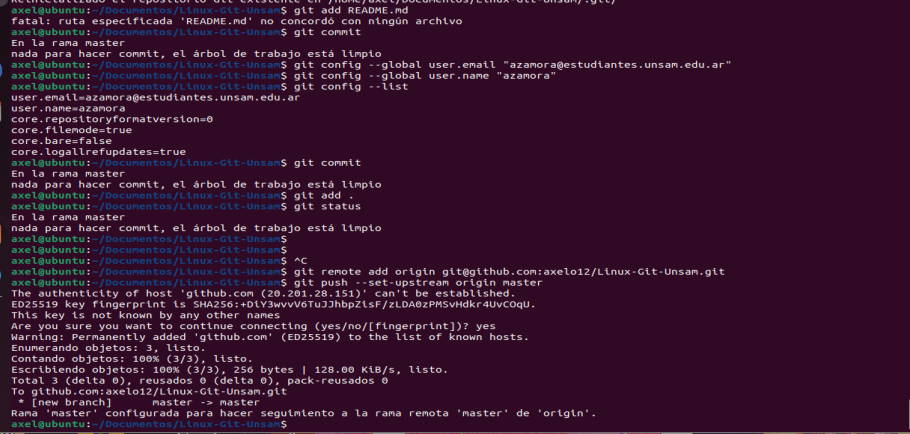
Este comando subirá los commits del branch “master” a su correspondiente branch remoto en "origin"[1](https://www.diegocmartin.com/tutorial-git/)[2](https://www.hostinger.es/tutoriales/comandos-de-git)[3](https://training.github.com/downloads/es_ES/github-git-cheat-sheet.pdf)[4](https://apiumhub.com/es/tech-blog-barcelona/20-comandos-basicos-de-git-que-todo-ingeniero-qa-debe-conocer/)[5](http://www.jesusclaramontegascon.com/vernoticias/principales-comandos-basicos-git-bash).

**Iniciar repositorio y subir archivo creado (la carpeta y el repositorio (github)tienen que tener el mismo nombre)**













Si estás trabajando en una rama por defecto (como "main" o "master"), es posible que necesites configurar la rama remota para hacer seguimiento de tu rama local usando:

git push --set-upstream origin nombre\_de\_la\_rama

Para crear una rama en Git y vincularla a GitHub, puedes seguir estos pasos[1](https://www.espai.es/blog/2021/05/como-trabajar-con-ramas-en-git-y-github/)[2](https://desarrolloweb.com/articulos/trabajar-ramas-git.html)[3](https://docs.github.com/es/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-and-deleting-branches-within-your-repository)[4](https://docs.github.com/es/desktop/contributing-and-collaborating-using-github-desktop/making-changes-in-a-branch/managing-branches-in-github-desktop):

1. **Crear una rama en Git**: Puedes usar el comando git branch nombre-de-la-rama para crear una nueva rama[1](https://www.espai.es/blog/2021/05/como-trabajar-con-ramas-en-git-y-github/). Para ver las ramas creadas, puedes usar el comando git branch[1](https://www.espai.es/blog/2021/05/como-trabajar-con-ramas-en-git-y-github/).
2. **Cambiar a la nueva rama**: Para cambiar a la nueva rama que acabas de crear, puedes usar el comando git checkout nombre-de-la-rama.
3. **Vincular la rama a GitHub**: Una vez que hayas hecho los cambios en tu rama y quieras subirla a GitHub, primero debes hacer un commit de tus cambios con git commit -m "tu mensaje". Luego, puedes usar el comando git push origin nombre-de-la-rama para subir tu rama a GitHub.

Si estás utilizando GitHub Desktop, puedes crear una nueva rama seleccionando “New Branch” (Nueva rama), escribiendo el nombre de la nueva rama y haciendo clic en “Create branch” (Crear rama)[4](https://docs.github.com/es/desktop/contributing-and-collaborating-using-github-desktop/making-changes-in-a-branch/managing-branches-in-github-desktop).

--------------------------------------------------------------------------------------------------------------------

BASH 102 – 1

Estos son algunos comandos de Bash que permiten ejecutar múltiples comandos y manipular variables:

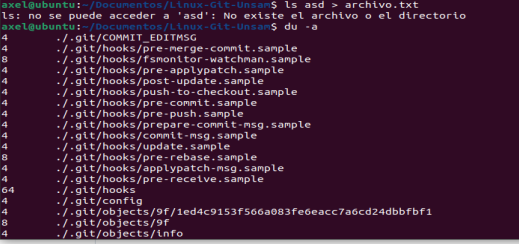
1. ls -l ; echo “hola”: Este comando ejecuta ls -l y luego echo “hola”[1](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C0%7Ca3078222-5132-4348-ad95-85594d6cec2b%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22%E2%97%8F%20Dos%20coma...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A28%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A3%2C%22highlightText%22%3A%22%E2%97%8F%20Dos%20comandos%3A%20ls%20-l%20%3B%20echo%20%E2%80%9Chola%E2%80%9D%22%2C%22snippets%22%3A%5B%5D%7D%7D). El punto y coma (;) permite ejecutar múltiples comandos en una sola línea.
2. ls -l & echo “hola”: Este comando ejecuta ls -l y echo “hola” en paralelo. El símbolo (&) permite ejecutar múltiples comandos al mismo tiempo.
3. ls unsam && echo “unsam existe”: Este comando ejecuta echo “unsam existe” solo si el comando ls unsam se ejecuta correctamente. Los dos símbolos de ampersand (&&) permiten ejecutar el segundo comando solo si el primero se ejecuta correctamente.
4. ls unsam || echo “unsam no existe”: Este comando ejecuta echo “unsam no existe” solo si el comando ls unsam falla. Los dos símbolos de barra vertical (||) permiten ejecutar el segundo comando solo si el primero falla.
5. VAR=no\_tengo\_java sh juego.sh: Este comando establece la variable VAR a “no\_tengo\_java” solo para la ejecución del script “juego.sh”. Esto es útil para pasar variables a scripts sin afectar el entorno actual.

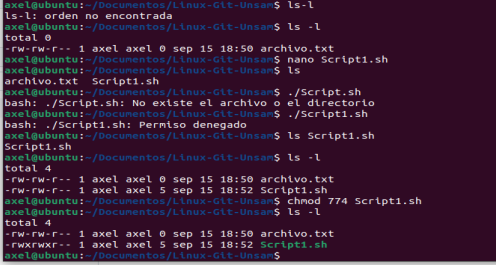
--------------------------------------------------------------------------------------------------------------------

BASH 102 – 2

Estás proporcionando información sobre las redirecciones en **BASH**, una característica muy útil que permite manipular la entrada y salida de los comandos:

1. **ls 1> archivo o ls > archivo**: Estos comandos redirigen la salida estándar (stdout) del comando ls a un archivo llamado “archivo”. Si el archivo ya existe, será sobrescrito.
2. **ls 1>> archivo o ls >> archivo**: Estos comandos redirigen la salida estándar del comando ls al final de un archivo llamado “archivo”. Si el archivo ya existe, la salida se añadirá al final del archivo.
3. **gcc main.c 2> errores**: Este comando redirige la salida de error (stderr) del comando gcc main.c a un archivo llamado “errores”. Esto es útil para guardar los mensajes de error para su posterior revisión.
4. **wc < entrada**: Este comando toma la entrada del comando wc desde un archivo llamado “entrada” en lugar de la entrada estándar (stdin). Esto es útil cuando quieres contar palabras, líneas o caracteres de un archivo específico.
5. **du -a | sort -n**: Este comando redirige la salida del comando du -a a la entrada del comando sort -n. El símbolo de tubería (|) permite conectar la salida de un comando a la entrada de otro, lo que es muy útil para crear flujos de trabajo complejos en la línea de comandos.





--------------------------------------------------------------------------------------------------------------------

BASH 102 – 3

Estás proporcionando información sobre la protección y sustitución en **BASH**:

1. **“”**: Las comillas dobles permiten proteger el contenido de una variable. Por ejemplo, VAR=perro;ls “\* $VAR” ejecutará ls \* perro, donde $VAR es reemplazado por su valor, en este caso, “perro”.
2. **‘ ‘**: Las comillas simples protegen la variable para que no se tome como tal. Por ejemplo, ls ‘$VAR’ buscará un archivo que se llame literalmente “$VAR”, sin reemplazar la variable[1](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C0%7Cd7a31bab-5930-4c2d-87d3-ad6121a1492c%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22%E2%97%8B%20VAR%3Dperr...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A30%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A4%2C%22highlightText%22%3A%22%E2%97%8B%20VAR%3Dperro%3Bls%20%E2%80%9C*%20%24VAR%E2%80%9D%20%2C%20da%20como%20resultado%20ls%20*%20perro%5Cr%5Cn%E2%97%8F%20%E2%80%98%20%E2%80%98%20protegen%20para%20que%20no%20se%20tome%20como%20variable%20%5Cr%5Cn%E2%97%8B%20ls%20%E2%80%98%24VAR%E2%80%99%20busca%20un%20archivo%20que%20se%20llame%20%24VAR%5Cr%5Cn%E2%97%8F%20%60%20%60%20permite%20que%20utilicemos%20la%20salida%20de%20un%20comando%20%5Cr%5Cn%E2%97%8B%20rm%20%60ls%20*c%60%22%2C%22snippets%22%3A%5B%5D%7D%7D).
3. : Las comillas invertidas permiten utilizar la salida de un comando como entrada para otro. Por ejemplo, rm ls \*c`` eliminará todos los archivos que terminen en “c”.
4. **()�()y(( ))**: Estos símbolos también permiten utilizar la salida de un comando como entrada para otro, pero $(( )) además permite realizar cálculos. Por ejemplo, echo $((5+3)) imprimirá el resultado de la suma 5+3, es decir, 8.

--------------------------------------------------------------------------------------------------------------------

BASH 102 – 4

Este es un script básico de Bash. Aquí está la explicación de cada línea:

1. #!/bin/bash: Esta es la línea shebang que le dice al sistema que debe usar Bash para interpretar el script.
2. # A sample Bash script: Este es un comentario. Los comentarios en Bash comienzan con un signo de número (#) y son ignorados por el intérprete de Bash.
3. echo Hello World: Este es un comando que imprime la cadena “Hello World” en la consola.

Por lo tanto, si ejecutas este script, verás “Hello World” impreso en tu consola.

--------------------------------------------------------------------------------------------------------------------

BASH 102 – 5

Estás proporcionando información sobre las pruebas en **BASH**, que son formas de evaluar condiciones:

1. **/bin/false y /bin/true**: Son comandos que siempre devuelven un estado falso y verdadero, respectivamente[1](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C0%7C1c80b610-182b-418b-a36e-6dc81aceb550%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22%E2%97%8F%20%2Fbin%2Ffal...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A32%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A4%2C%22highlightText%22%3A%22%E2%97%8F%20%2Fbin%2Ffalse%20devuelve%20siempre%20falso%2C%20%2Fbin%2Ftrue%20devuelve%20siempre%20verdadero%22%2C%22snippets%22%3A%5B%5D%7D%7D).
2. **[ -e archivo ]**: Esta prueba verifica si un archivo existe.
3. **[ -d archivo ]**: Esta prueba verifica si un archivo es un directorio.
4. **[ arch1 -nt arch2 ]**: Esta prueba verifica si el archivo “arch2” es más reciente que “arch1”.
5. **[ aaa == aaaa ]**: Esta prueba compara dos cadenas de texto. En este caso, devuelve falso porque “aaa” no es igual a “aaaa”.

Estas pruebas son útiles en scripts de Bash para controlar el flujo de ejecución basado en ciertas condiciones.

--------------------------------------------------------------------------------------------------------------------

BASH 103 – 1

Estás proporcionando información sobre las pruebas lógicas en **BASH**:

1. **IF**: Se utiliza para evaluar una condición. Si la condición es verdadera, entonces se ejecuta el comando dentro del bloque if. Aquí hay un ejemplo:
2. if [ 200 -gt 100 ]
3. then
4. echo “es mayor”
5. fi

En este ejemplo, la condición [ 200 -gt 100 ] evalúa si 200 es mayor que 100. Como esto es verdadero, se ejecuta el comando echo “es mayor”, y por lo tanto se imprime “es mayor” en la consola.

Las condiciones posibles para las pruebas en BASH incluyen: = (igual), != (distinto), -ge (mayor o igual), -e (existe), -le (menor o igual), -z (es vacío?), -n (es no vacío?), entre otras.

Estás proporcionando información sobre las condiciones posibles en **BASH**:

1. **= (igual)**: Esta condición verifica si dos cadenas o números son iguales. Por ejemplo, [ "abc" = "abc" ] es verdadero, pero [ 1 = 2 ] es falso.
2. **!= (distinto)**: Esta condición verifica si dos cadenas o números son diferentes. Por ejemplo, [ "abc" != "def" ] es verdadero, pero [ 1 != 1 ] es falso.
3. **-ge (mayor o igual)**: Esta condición verifica si un número es mayor o igual que otro. Por ejemplo, [ 2 -ge 1 ] es verdadero, pero [ 1 -ge 2 ] es falso.
4. **-e (existe)**: Esta condición verifica si un archivo existe. Por ejemplo, [ -e "/path/to/file" ] es verdadero si el archivo existe, y falso si no existe.
5. **-le (menor o igual)**: Esta condición verifica si un número es menor o igual que otro. Por ejemplo, [ 1 -le 2 ] es verdadero, pero [ 2 -le 1 ] es falso.
6. **-z (es vacío?)**: Esta condición verifica si una cadena está vacía. Por ejemplo, [ -z "" ] es verdadero, pero [ -z "no vacío" ] es falso.
7. **-n (es no vacío?)**: Esta condición verifica si una cadena no está vacía. Por ejemplo, [ -n "no vacío" ] es verdadero, pero [ -n "" ] es falso.

--------------------------------------------------------------------------------------------------------------------

BASH 103 – 2

case: como un switch en C

case "$1" in

si)

echo "Ok, esta aprobado"

;;

no)

echo "Por qué desaprobe???"

;;

\*)

echo "Si o no son las unicas respuestas validas!"

;;

esac

El fragmento de código que has seleccionado es un ejemplo de una estructura de control case en Bash, que es similar a la instrucción switch en C. Aquí está la explicación:

* case "$1" in: Esta línea comienza la estructura de control case, evaluando el valor de la variable $1 (el primer argumento pasado al script).
* si) echo "Ok, esta aprobado" ;;: Si $1 es igual a “si”, entonces se imprime "Ok, esta aprobado"[1](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C0%7C3bb0b95e-bb8c-43d0-a936-4e51286a5971%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22echo%20%5C%22Ok%2C%20...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A34%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A6%2C%22highlightText%22%3A%22echo%20%5C%22Ok%2C%20esta%20aprobado%5C%22%22%2C%22snippets%22%3A%5B%5D%7D%7D).
* no) echo "Por qué desaprobe???"[^2^][2] ;;: Si $1 es igual a “no”, entonces se imprime “Por qué desaprobe???”.
* \*) echo "Si o no son las unicas respuestas validas!"[^3^][3] ;;: Esta es la opción por defecto que se ejecuta si ninguna de las condiciones anteriores se cumple. En este caso, se imprime "Si o no son las unicas respuestas validas!"[3](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C2%7C3bb0b95e-bb8c-43d0-a936-4e51286a5971%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22echo%20%5C%22Si%20o...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A34%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A12%2C%22highlightText%22%3A%22echo%20%5C%22Si%20o%20no%20son%20las%20unicas%20respuestas%20validas!%5C%22%22%2C%22snippets%22%3A%5B%5D%7D%7D).
* esac: Esta línea finaliza la estructura de control case.

Por lo tanto, este script toma un argumento y verifica si es “si” o “no”, e imprime un mensaje en consecuencia. Si el argumento es cualquier otra cosa, imprime un mensaje de error.

--------------------------------------------------------------------------------------------------------------------

BASH 103 – 3

Logicos

* while: mismo tipo de construcción que el if

i=0

while [ $i -lt 10 ]

do

echo $i

i=$((i + 1))

done

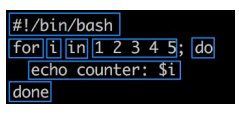
El fragmento de código que seleccionaste es un ejemplo de un bucle while en Bash. Aquí está la explicación:

* i=0: Esta línea inicializa la variable i en 0.
* while [ $i -lt 10 ]: Este es el inicio del bucle while[1](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C0%7Cc77628e3-6412-4343-910b-6802f91f49dd%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22while%20%5B%20%24i...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A35%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A6%2C%22highlightText%22%3A%22while%20%5B%20%24i%20-lt%2010%20%5D%22%2C%22snippets%22%3A%5B%5D%7D%7D). El bucle continuará mientras la condición $i -lt 10 (es decir, i es menor que 10) sea verdadera.
* do: Esta palabra clave indica el inicio del cuerpo del bucle.
* echo $i: Este comando imprime el valor actual de i.
* i=$((i + 1)): Esta línea incrementa el valor de i en 1[2](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C1%7Cc77628e3-6412-4343-910b-6802f91f49dd%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22i%3D%24((i%20%2B%201...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A35%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A9%2C%22highlightText%22%3A%22i%3D%24((i%20%2B%201))%22%2C%22snippets%22%3A%5B%5D%7D%7D).
* done: Esta palabra clave indica el final del cuerpo del bucle.

Por lo tanto, este script imprimirá los números del 0 al 9 en la consola.

--------------------------------------------------------------------------------------------------------------------

BASH 103 – 4



Logicos

* for: no utiliza condición, si no, una lista

for i in \*

do

echo "archivo $i"

done

**El fragmento de código que seleccionaste es un ejemplo de un bucle for en Bash. Aquí está la explicación:**

* for i in \*: Este es el inicio del bucle for. El bucle iterará sobre todos los archivos y directorios en el directorio actual. La variable i tomará el valor de cada archivo o directorio por turno.
* do: Esta palabra clave indica el inicio del cuerpo del bucle.
* echo "archivo $i": Este comando imprime la cadena "archivo " seguida del nombre del archivo o directorio actual.
* done: Esta palabra clave indica el final del cuerpo del bucle.

Por lo tanto, este script imprimirá una lista de todos los archivos y directorios en el directorio actual, precedidos por la palabra "archivo ".

--------------------------------------------------------------------------------------------------------------------

BASH 103 – 5

Variables

* Para recibir en un script bash una opción escrita por consola:

○ $1 , $2 , $3, ${10}

* Para recibir todos las opciones en una lista

○ $\* (todos concatenados) ○ $@ (separados como en un array)

#./script.sh hola

$1 seria igual a hola

**Estás proporcionando información sobre cómo manejar las variables en un script de Bash:**

1. **$1 , $2 , 3,3,{10}**: Estas son variables especiales en Bash que representan los argumentos pasados a un script. $1 es el primer argumento, $2 es el segundo argumento, y así sucesivamente. ${10} representa el décimo argumento.
2. **∗�∗y@**: Estas son variables especiales en Bash que representan todos los argumentos pasados a un script. $\* devuelve todos los argumentos como una sola cadena, mientras que $@ devuelve todos los argumentos como una lista separada.

Por ejemplo, si ejecutas ./script.sh hola, entonces dentro del script, $1 sería igual a “hola”.

--------------------------------------------------------------------------------------------------------------------

El directorio /proc

[proc(5): process info pseudo-file system - Linux man page (die.net)](https://linux.die.net/man/5/proc)

El directorio /proc en Linux es un sistema de archivos virtual que contiene información sobre el sistema y los procesos que se están ejecutando. Aquí te explico algunos de los archivos que mencionaste:

1. **/proc/PID/cwd**: Este archivo es un enlace simbólico al directorio de trabajo actual del proceso con el ID de proceso (PID) especificado[1](https://www.linuxtotal.com.mx/index.php?cont=info_admon_016).
2. **/proc/PID/environ**: Este archivo contiene las variables de entorno para el proceso con el PID especificado[1](https://www.linuxtotal.com.mx/index.php?cont=info_admon_016).
3. **/proc/PID/status**: Este archivo proporciona información sobre el estado del proceso con el PID especificado[1](https://www.linuxtotal.com.mx/index.php?cont=info_admon_016).
4. **/proc/cpuinfo**: Este archivo proporciona información detallada sobre la CPU, incluyendo su tipo, marca, modelo y características[1](https://www.linuxtotal.com.mx/index.php?cont=info_admon_016).
5. **/proc/filesystems**: Este archivo enumera todos los sistemas de archivos que actualmente reconoce el kernel[1](https://www.linuxtotal.com.mx/index.php?cont=info_admon_016).
6. **echo 1 > /proc/sys/net/ipv4/ip\_forward**: Este comando habilita el reenvío de paquetes IP en el kernel, lo que permite que la máquina funcione como un router[1](https://www.linuxtotal.com.mx/index.php?cont=info_admon_016).

**Aquí tienes algunos ejemplos de cómo usar el directorio /proc en Linux:**

1. **Ver la versión del kernel**: Puedes usar el archivo /proc/version para ver la versión del kernel que estás ejecutando. Para hacer esto, puedes usar el comando cat /proc/version[1](https://www.linuxtotal.com.mx/index.php?cont=info_admon_016).
2. **Ver los dispositivos configurados**: El archivo /proc/devices muestra una lista de los controladores de dispositivos que están configurados en el kernel que está en ejecución[2](https://tldp.org/pub/Linux/docs/ldp-archived/system-admin-guide/translations/es/html/ch04s07.html).
3. **Ver los sistemas de archivos soportados**: El archivo /proc/filesystems muestra una lista de los sistemas de archivos que están soportados por el kernel[2](https://tldp.org/pub/Linux/docs/ldp-archived/system-admin-guide/translations/es/html/ch04s07.html).
4. **Ver información sobre un proceso**: Cada proceso tiene su propio subdirectorio en /proc con su ID de proceso (PID) como nombre. Puedes buscar información sobre un proceso específico mirando en su subdirectorio[3](https://laboratoriolinux.es/index.php/-noticias-mundo-linux-/software/29918-conoce-el-uso-del-directorio-proc.html). Por ejemplo, puedes usar ls /proc/1234 para ver los archivos y directorios relacionados con el proceso con PID 1234.
5. **Habilitar el reenvío de paquetes IP**: Puedes habilitar el reenvío de paquetes IP en el kernel (lo que permite que tu máquina funcione como un router) escribiendo 1 en el archivo /proc/sys/net/ipv4/ip\_forward. Para hacer esto, puedes usar el comando echo 1 > /proc/sys/net/ipv4/ip\_forward.

Por favor, ten en cuenta que necesitarás permisos de root para leer o escribir en muchos de estos archivos

--------------------------------------------------------------------------------------------------------------------

**Aplicaciones de Servidores Linux populares**

Servidor Web: HTTP/HTTPS .

* Apache
* Nginx

Servidor de Correo: MTA (SMTP) , MSA, POP/IMAP

* Postfix
* Sendmail
* Dovecot

Servidor de Resolución de Nombres: DNS

* Bind

Servidor de Directorio: openLDAP

Servidor de Base de Datos: SQL o NoSQL

* Mysql
* Postgresql
* MongoDB

**Estás proporcionando información sobre varios tipos de servidores populares en Linux:**

1. **Servidor Web**: Estos servidores manejan las solicitudes HTTP/HTTPS[1](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C0%7Ced088951-7b67-4877-821b-7e567eebd6ec%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22Aplicacion...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A39%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A1%2C%22highlightText%22%3A%22Aplicaciones%20de%20Servidores%20Linux%20populares%5Cr%5Cn%E2%97%8F%20Servidor%20Web%3A%20HTTP%2FHTTPS%20.%22%2C%22snippets%22%3A%5B%5D%7D%7D). Los ejemplos más comunes son Apache y Nginx.
2. **Servidor de Correo**: Estos servidores manejan los protocolos de correo como SMTP, MSA, POP/IMAP[2](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C1%7Ced088951-7b67-4877-821b-7e567eebd6ec%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22%E2%97%8F%20Servidor...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A39%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A5%2C%22highlightText%22%3A%22%E2%97%8F%20Servidor%20de%20Correo%3A%20MTA%20(SMTP)%20%2C%20MSA%2C%20POP%2FIMAP%5Cr%5Cn%E2%97%8B%20Postfix%22%2C%22snippets%22%3A%5B%5D%7D%7D). Algunos ejemplos son Postfix, Sendmail y Dovecot.
3. **Servidor de Resolución de Nombres**: Estos servidores manejan las solicitudes DNS[3](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C2%7Ced088951-7b67-4877-821b-7e567eebd6ec%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22%E2%97%8F%20Servidor...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A39%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A9%2C%22highlightText%22%3A%22%E2%97%8F%20Servidor%20de%20Resoluci%C3%B3n%20de%20Nombres%3A%20DNS%22%2C%22snippets%22%3A%5B%5D%7D%7D). Un ejemplo común es Bind.
4. **Servidor de Directorio**: Estos servidores proporcionan servicios de directorio como openLDAP[4](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C3%7Ced088951-7b67-4877-821b-7e567eebd6ec%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22%E2%97%8F%20Servidor...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A39%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A11%2C%22highlightText%22%3A%22%E2%97%8F%20Servidor%20de%20Directorio%3A%20openLDAP%22%2C%22snippets%22%3A%5B%5D%7D%7D).
5. **Servidor de Base de Datos**: Estos servidores manejan las bases de datos SQL o NoSQL[5](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C4%7Ced088951-7b67-4877-821b-7e567eebd6ec%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22%E2%97%8F%20Servidor...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A39%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A12%2C%22highlightText%22%3A%22%E2%97%8F%20Servidor%20de%20Base%20de%20Datos%3A%20SQL%20o%20NoSQL%22%2C%22snippets%22%3A%5B%5D%7D%7D). Algunos ejemplos son MySQL, PostgreSQL y MongoDB.

Cada uno de estos servidores tiene un propósito específico y se utiliza para diferentes aplicaciones en un sistema Linux.

--------------------------------------------------------------------------------------------------------------------

Escritorio Remoto (VNC)

Instalar el servidor

* sudo apt install x11vnc

Configuración de password

* sudo x11vnc -storepasswd /etc/password\_vnc

Correr el comando

* /usr/bin/x11vnc -forever -loop -noxdamage -repeat -rfbauth /etc/password\_vnc -rfbport 5900 -shared

**Estás proporcionando información sobre cómo configurar un servidor x11vnc en Linux:**

1. [**Instalación del servidor**: El comando sudo apt install x11vnc instala el servidor x11vnc en tu sistema1](https://devicetests.com/setup-x11vnc-remote-access-ubuntu-server).
2. [**Configuración de la contraseña**: El comando sudo x11vnc -storepasswd /etc/password\_vnc configura una contraseña para el servidor x11vnc](about:blank)[1](https://devicetests.com/setup-x11vnc-remote-access-ubuntu-server).
3. [**Ejecución del servidor**: El comando /usr/bin/x11vnc -forever -loop -noxdamage -repeat -rfbauth /etc/password\_vnc -rfbport 5900 -shared inicia el servidor x11vnc con varias opciones1](https://devicetests.com/setup-x11vnc-remote-access-ubuntu-server):
   * -forever: Hace que el servidor se ejecute indefinidamente.
   * -loop: Reinicia el servidor si se desconecta.
   * -noxdamage: Desactiva la extensión XDamage para mejorar la compatibilidad.
   * -repeat: Permite que las teclas se repitan en el cliente VNC.
   * -rfbauth /etc/password\_vnc: Usa el archivo de contraseña especificado para la autenticación.
   * -rfbport 5900: Establece el puerto del servidor VNC en 5900.
   * -shared: Permite múltiples conexiones al servidor.

[Por lo tanto, este script instala, configura y ejecuta un servidor x11vnc en tu sistema](about:blank)[1](https://devicetests.com/setup-x11vnc-remote-access-ubuntu-server).

--------------------------------------------------------------------------------------------------------------------

SSH y SCP

Nombre del protocolo y del programa que lo implementa.

* Sirve para acceder remotamente a máquinas
* Arma un túnel seguro sobre redes inseguras.

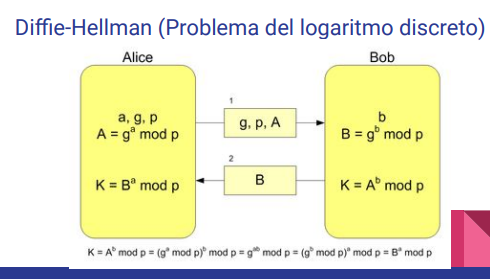
Instalacion

* sudo apt install ssh

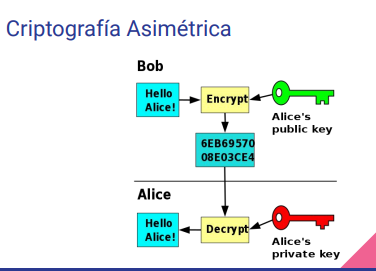
Configuracion

* archivo /etc/ssh/sshd\_config

--------------------------------------------------------------------------------------------------------------------



--------------------------------------------------------------------------------------------------------------------



--------------------------------------------------------------------------------------------------------------------

SSH keys y SSH config

Manera de acceder a los equipos sin tener que escribir la password. Más seguridad y automatización.

ssh-keygen -t rsa -b 4096 -C "mail@unsam.edu.ar"

Generating public/private rsa key pair.

* ssh-copy-id
* ~/.ssh/
* ~/.ssh/config

Host vpn

Hostname vpn.claro.com.ar

IdentityFile ~/.ssh/id\_rsa

user root

**El fragmento de código que seleccionaste es un ejemplo de cómo configurar y utilizar SSH (Secure Shell) para la autenticación basada en claves en Linux:**

1. [ssh-keygen -t rsa -b 4096 -C “mail@unsam.edu.ar”: Este comando genera un nuevo par de claves SSH de tipo RSA con una longitud de 4096 bits](about:blank)[1](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C0%7C64925730-eb64-49aa-a618-0fdc50706229%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22ssh-keygen...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A45%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A4%2C%22highlightText%22%3A%22ssh-keygen%20-t%20rsa%20-b%204096%20-C%20%5C%22mail%40unsam.edu.ar%5C%22%5Cr%5CnGenerating%20public%2Fprivate%20rsa%20key%20pair.%22%2C%22snippets%22%3A%5B%5D%7D%7D). El correo electrónico “mail@unsam.edu.ar” se utiliza como un comentario para identificar la clave.
2. ssh-copy-id: Este comando se utiliza para instalar tu clave pública en el servidor SSH. Después de copiar la clave, puedes iniciar sesión en el servidor sin contraseña.
3. ~/.ssh/: Este es el directorio donde se almacenan las claves SSH y otros archivos relacionados con SSH en tu sistema local.
4. ~/.ssh/config: Este archivo se utiliza para configurar ajustes de SSH a nivel de usuario, como alias para nombres de host, usuarios y puertos por defecto, y muchas otras opciones.
5. [Host vpn Hostname vpn.claro.com.ar](about:blank)[2](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C1%7C64925730-eb64-49aa-a618-0fdc50706229%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22Hostname%20v...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A45%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A10%2C%22highlightText%22%3A%22Hostname%20vpn.claro.com.ar%22%2C%22snippets%22%3A%5B%5D%7D%7D) [IdentityFile ~/.ssh/id\_rsa](about:blank)[3](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C2%7C64925730-eb64-49aa-a618-0fdc50706229%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22IdentityFi...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A45%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A11%2C%22highlightText%22%3A%22IdentityFile%20~%2F.ssh%2Fid_rsa%22%2C%22snippets%22%3A%5B%5D%7D%7D) user root: Este es un ejemplo de una configuración de host en el archivo ~/.ssh/config. Define un alias de host llamado “vpn” para el nombre de host “vpn.claro.com.ar”, especifica que se debe usar la clave privada almacenada en ~/.ssh/id\_rsa para la autenticación, y establece “root” como el nombre de usuario por defecto para este host.

--------------------------------------------------------------------------------------------------------------------

IPTables – Netfilter

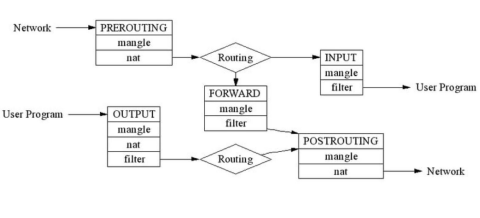
Hay cinco tablas ya incorporadas, cada una de ellas contiene ciertas cadenas predefinidas:

Tablas

* raw
* filter (default)
* nat
* mangle
* security

**Las tablas en iptables son conjuntos de cadenas que contienen reglas incorporadas o definidas por el usuario. Cuando un paquete llega, atraviesa las cadenas y se compara con las reglas que contiene.** [**Si una regla coincide, el paquete es tratado según la especificación de la regla1**](https://techlib.net/techedu/iptables/)**. Aquí te explico cada una de las tablas que mencionaste:**

1. **raw**: Esta tabla se utiliza principalmente para configurar excepciones para el seguimiento de conexiones.
2. **filter (default)**: Esta es la tabla predeterminada y probablemente la más utilizada. Se utiliza para filtrar paquetes.
3. **nat**: Esta tabla se utiliza para la traducción de direcciones de red (Network Address Translation - NAT).
4. **mangle**: Esta tabla se utiliza para la alteración especial de paquetes.
5. **security**: Esta tabla se utiliza para configurar políticas de control de acceso obligatorio (Mandatory Access Control - MAC), como SELinux



--------------------------------------------------------------------------------------------------------------------

IPTables – Netfilter

-t : Selecciona la tabla, por defecto trabaja con filter

-L : Muestra las reglas activas

-v : Muestra en modo verboso

-n : Muestra en formato numérico

-F : Borra todas las reglas de la tabla

-Z : Pone los contadores en cero

-P : Cambia la Política de la CADENA

-A : Agrega una regla al final de la CADENA

-I : Inserta al principio una regla en la CADENA

-p : Indica el protocolo con el cual se va a trabajar

-m : Incorpora un módulo

-s : Determina el origen del paquete (IP)

-d : Determina el destino del paquete (IP)

-j : Indica la acción a realizar

Acciones terminantes: ACCEPT, DROP, REJECT

--------------------------------------------------------------------------------------------------------------------

IPTables – Netfilter

# Abrir el puerto TCP/80 para todos

iptables -A INPUT -p tcp --dport 80 -j ACCEPT

# Descartar el tráfico que venga desde 3.3.3.3

iptables -A INPUT -s 3.3.3.3 -j DROP

# Aceptar paquetes de conexiones ya establecidas

iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Escribir logs

iptables -A INPUT -p tcp -j LOG --log-prefix "INPUT packets"

# Listar reglas configuradas

iptables -L -n -v

# Cambiar Política

iptables -P INPUT DROP

**Estás proporcionando información sobre cómo usar iptables, una herramienta de filtrado de paquetes en Linux:**

1. iptables -A INPUT -p tcp --dport 80 -j ACCEPT: Este comando abre el puerto TCP 80 para todos.
2. [iptables -A INPUT -s 3.3.3.3 -j DROP: Este comando descarta todo el tráfico que proviene de la dirección IP 3.3.3.3](about:blank)[1](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C0%7C7a642597-4dc6-4491-94e5-a02f76953e89%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22iptables%20-...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A49%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A3%2C%22highlightText%22%3A%22iptables%20-A%20INPUT%20-p%20tcp%20--dport%2080%20-j%20ACCEPT%5Cr%5Cn%23%20Descartar%20el%20tr%C3%A1fico%20que%20venga%20desde%203.3.3.3%20%5Cr%5Cniptables%20-A%20INPUT%20-s%203.3.3.3%20-j%20DROP%22%2C%22snippets%22%3A%5B%5D%7D%7D).
3. [iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT: Este comando acepta paquetes de conexiones ya establecidas](about:blank)[2](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C1%7C7a642597-4dc6-4491-94e5-a02f76953e89%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22iptables%20-...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A50%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A7%2C%22highlightText%22%3A%22iptables%20-A%20INPUT%20-m%20state%20--state%20ESTABLISHED%2CRELATED%20-j%20ACCEPT%22%2C%22snippets%22%3A%5B%5D%7D%7D).
4. [iptables -A INPUT -p tcp -j LOG --log-prefix “INPUT packets”: Este comando registra los paquetes TCP entrantes con el prefijo “INPUT packets” en el log del sistema](about:blank)[3](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C2%7C7a642597-4dc6-4491-94e5-a02f76953e89%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22iptables%20-...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A49%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A10%2C%22highlightText%22%3A%22iptables%20-A%20INPUT%20-p%20tcp%20-j%20LOG%20--log-prefix%20%5C%22INPUT%20%5Cr%5Cnpackets%5C%22%22%2C%22snippets%22%3A%5B%5D%7D%7D).
5. iptables -L -n -v: Este comando muestra las reglas de iptables activas en formato numérico y modo verboso.
6. iptables -P INPUT DROP: Este comando cambia la política predeterminada para las conexiones entrantes a DROP, lo que significa que se descartarán todas las conexiones entrantes que no coincidan con ninguna regla.

--------------------------------------------------------------------------------------------------------------------

**IPTables - Netfilter - Host Personal**

iptables -X

iptables -F

iptables -Z

iptables -P INPUT DROP

iptables -P OUTPUT ACCEPT

iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

Iptables -A INPUT -p udp --dport 53 -j ACCEPT

**Estás proporcionando información sobre cómo usar iptables, una herramienta de filtrado de paquetes en Linux:**

1. iptables -X: Este comando borra todas las cadenas definidas por el usuario.
2. [iptables -F: Este comando borra todas las reglas de la tabla](about:blank)[1](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C0%7C1664ec6a-39e7-4799-87b0-929a824809a5%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22-F%20%3A%20Borra...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A48%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A6%2C%22highlightText%22%3A%22-F%20%3A%20Borra%20todas%20las%20reglas%20de%20la%20tabla%22%2C%22snippets%22%3A%5B%5D%7D%7D).
3. [iptables -Z: Este comando pone los contadores en cero](about:blank)[2](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C1%7C1664ec6a-39e7-4799-87b0-929a824809a5%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22-Z%20%3A%20Pone%20...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A48%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A7%2C%22highlightText%22%3A%22-Z%20%3A%20Pone%20los%20contadores%20en%20cero%22%2C%22snippets%22%3A%5B%5D%7D%7D).
4. iptables -P INPUT DROP: Este comando cambia la política predeterminada para las conexiones entrantes a DROP, lo que significa que se descartarán todas las conexiones entrantes que no coincidan con ninguna regla.
5. iptables -P OUTPUT ACCEPT: Este comando cambia la política predeterminada para las conexiones salientes a ACCEPT, lo que significa que se permitirán todas las conexiones salientes a menos que coincidan con una regla que indique lo contrario.
6. [iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT: Este comando permite paquetes de conexiones ya establecidas](about:blank)[3](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C2%7C1664ec6a-39e7-4799-87b0-929a824809a5%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22iptables%20-...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A50%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A7%2C%22highlightText%22%3A%22iptables%20-A%20INPUT%20-m%20state%20--state%20ESTABLISHED%2CRELATED%20-j%20ACCEPT%22%2C%22snippets%22%3A%5B%5D%7D%7D)[4](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C3%7C1664ec6a-39e7-4799-87b0-929a824809a5%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22Iptables%20-...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A51%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A8%2C%22highlightText%22%3A%22Iptables%20-A%20OUTPUT%20-m%20state%20--state%20ESTABLISHED%2CRELATED%20-j%20ACCEPT%22%2C%22snippets%22%3A%5B%5D%7D%7D).
7. [Iptables -A INPUT -p udp --dport 53 -j ACCEPT: Este comando permite el tráfico UDP entrante al puerto 53](about:blank)[5](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,channelstable,&shellsig=47bd0c55e5150d62fb3c9ba5b39c53ba98dedffb&setlang=es-419&darkschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C4%7C1664ec6a-39e7-4799-87b0-929a824809a5%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22Iptables%20-...%22%2C%22pageType%22%3A%22pdf%22%2C%22pageIndex%22%3A51%2C%22relatedPageUrl%22%3A%22file%253A%252F%252F%252FC%253A%252FUsers%252Fzamor%252FDesktop%252F01%252520-%252520Administracion%252520de%252520Sistemas%252520GNU_Linux%252520(2).pdf%22%2C%22lineIndex%22%3A12%2C%22highlightText%22%3A%22Iptables%20-A%20OUTPUT%20-p%20udp%20--dport%2053%20-d%208.8.8.8%20-j%20ACCEPT%22%2C%22snippets%22%3A%5B%5D%7D%7D).

Por lo tanto, este script configura iptables para permitir solo cierto tráfico y descartar todo lo demás.

--------------------------------------------------------------------------------------------------------------------

**IPTables - Netfilter - Server Web**

iptables -X

iptables -F

iptables -Z

Iptables -P INPUT DROP

Iptables -P OUTPUT DROP

Iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

Iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

Iptables -A INPUT -p -tcp --dport 443 -j ACCEPT

Iptables -A INPUT -p tcp --dport 22 -s 192.168.5.5 -j ACCEPT

Iptables -A INPUT -p icmp -j ACCEPT

Iptables -A OUTPUT -p udp --dport 53 -d 8.8.8.8 -j ACCEPT

**Estás proporcionando información sobre cómo configurar iptables, una herramienta de filtrado de paquetes en Linux:**

1. iptables -X: Este comando borra todas las cadenas definidas por el usuario.
2. iptables -F: Este comando borra todas las reglas de la tabla.
3. iptables -Z: Este comando pone los contadores en cero.
4. Iptables -P INPUT DROP: Este comando cambia la política predeterminada para las conexiones entrantes a DROP, lo que significa que se descartarán todas las conexiones entrantes que no coincidan con ninguna regla.
5. Iptables -P OUTPUT DROP: Este comando cambia la política predeterminada para las conexiones salientes a DROP, lo que significa que se descartarán todas las conexiones salientes que no coincidan con ninguna regla.
6. Iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT: Este comando permite paquetes de conexiones ya establecidas.
7. Iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT: Este comando permite paquetes de conexiones ya establecidas.
8. Iptables -A INPUT -p tcp --dport 443 -j ACCEPT: Este comando permite el tráfico TCP entrante al puerto 443.
9. Iptables -A INPUT -p tcp --dport 22 -s 192.168.5.5 -j ACCEPT: Este comando permite el tráfico TCP entrante al puerto 22 solo desde la dirección IP 192.168.5.5.
10. Iptables -A INPUT -p icmp -j ACCEPT: Este comando permite el tráfico ICMP entrante.
11. Iptables -A OUTPUT -p udp --dport 53 -d 8.8.8.8 -j ACCEPT: Este comando permite el tráfico UDP saliente al puerto 53 destinado a la dirección IP 8.8.8.8.

Por lo tanto, este script configura iptables para permitir solo cierto tráfico y descartar todo lo demás.

--------------------------------------------------------------------------------------------------------------------