

LINUX

Sumario

Administrar texto.....	2
Awk:.....	2
Base64:.....	3
Cut:.....	3
Diff:.....	3
Grep:.....	3
Head y Tail:.....	4
Rev:.....	4
Sed:.....	4
Sort:.....	4
Strings:.....	4
Uniq:.....	5
Tr:.....	5
Wc:.....	5
Alias.....	5
Atajos.....	5
Directorios y archivos.....	6
Compresión:.....	6
Creado y borrado:.....	7
File:.....	7
Find:.....	7
Listado:.....	8
Crontab:.....	8
Espacio en disco.....	9
Enlaces.....	9
Gestión de usuarios.....	10
GRUB.....	10
Config GRUB:.....	10
Reseteo RedHat 8:.....	11
LDAP.....	11
Firewall y puertos.....	11
Firewalld:.....	11
UFW:.....	12
Permisos.....	13
Chmod/own:.....	13
ACL (Access Control List):.....	14
Umask:.....	14
Atributos de fichero:.....	14
Procesos.....	15
Ps:.....	15
Wait:.....	16
Servicios.....	16
Red y parámetros del sistema.....	17
SSH:.....	17
FTP:.....	18
Uname:.....	18

Traceroute:.....	18
Ping:.....	18
Otros:.....	18
Directorios y archivos importantes.....	19
Volúmenes y gestión de discos:.....	21
NFS:.....	22
LVM:.....	23
VDO/STRATIS:.....	27
Redireccionamiento.....	28
Vim:.....	29
Tmux.....	29
Otros.....	30
Instalación.....	31
RedHat, Fedora o CenOS:.....	31
Repositorios RedHat:.....	32
Bash Scripting.....	32

Puntos en rojo repasar para RHCSA

Administrar texto

Awk:

Sirve para mostrar un texto específico de salida.

Para mostrar la línea que queramos. Ejemplo: `awk "NR==3"`

`awk '{print $1}'` : Me muestra el primer argumento de cada línea

`awk 'NF{print $NF}'` : Me muestra el último argumento de cada línea

Con `substr(cadena, inicio, longitud)` extraemos una subcadena de texto. Es decir podemos coger las letras que queramos de cada palabra.

Ejemplo:

```
awk '{for(i=1;i<=NF;i++) print substr($i,1,1)}' | tr -d '\n'
```

Recorremos cada palabra con el for y luego con el substr elegimos la primera letra de cada palabra, luego quitamos los espacios.

Awk es muy extenso y tiene muchas más funciones.

Base64:

Sirve para codificar y decodificar archivos en base64

Codificar palabras: `echo -n "palabra" | base64`

Codificar archivo: `base64 "archivo"`

Decodificar archivo: `base64 --decode archivo.txt`

Cut:

Me corta la salida de un comando, se usa siempre con una opción, tiene usos muy útiles:

Ejemplo: `cut -d ":" -f 1,2,3 texto.txt`

Le indicamos el delimitador con `-d ":"` y que muestre las columnas 1,2 y 3 con `-f`.

Podríamos mezclarlo con sort para que nos ordene la salida:

`cut -d ":" -f 1,2,3 texto.txt | sort`

Podríamos mezclarlo con grep para que nos busque solo ciertos nombres:

`cut -d ":" -f 1,2,3 texto.txt | grep Paco | sort.`

Diff:

Sirve para diferenciar dos archivos.

Grep:

Filtra por un patrón de caracteres y despliega su contenido.

Podemos usarlo junto con ls para listar ficheros que contengan una palabra en su nombre:

`ls -lia | grep script:` Me listarían todos los ficheros que tienen script en su nombre.

Para no tener que hacer `cat data.txt | grep "hola"`, podemos hacer `grep "hola" data.txt` en archivos.

Añadiendo `^` obligamos a que la palabra empiece por eso, con `$` le decimos que termine con eso

Opciones:

`-A`: Te muestra x líneas por debajo de lo que digas (`B` para arriba)

`-c`: Número de líneas que contiene el patrón.

`-E`: Filtrado múltiple. Ejemplo: `grep -E "Entrada total|Salida total"`

`-i`: Te muestra las mayúsculas y minúsculas

`-n`: Muestra las líneas que coinciden y su número.

- l: En una carpeta, te dice los ficheros que contienen ese patrón.
- r: Busca recursivamente en subdirectorios.
- s: Suprime errores (se suele usar para subdirectorios)
- v: Te muestra las líneas que no contienen lo que pongas

Head y Tail:

Me muestra las primeras o últimas líneas

- n: Indicamos las líneas que queremos (con num negativo quitamos las últimas o primeras)
- f: Seguimos monitorizando el archivo, útil para el tail en logs para ver que va escribiendo.

Ej: `cat archivo.txt | head -n -2` (Nos quita las dos últimas líneas)

Ej: `tail -f /var/log/messages`

Rev:

Sirve para revertir una cadena de texto.

Sed:

Sirve para modificar texto normalmente, lo más usado es con s:

`sed 's/perro/gato'` Con g al final me lo aplica a toda la línea
(Normalmente es solo al primer resultado)

`sed "3d" distros-deb.txt > distros-deb-ok.txt` Borra la línea 3

Si el sed es con variables se pone con comillas dobles.

Sort:

Sirve para ordenar la salida de un comando:

Por ejemplo un texto que tenga las líneas desordenadas con `sort texto.txt` mostramos las líneas ordenadas alfabéticamente. También podemos hacerlo con varios archivos a la vez:

`sort texto.txt texto2.txt`

Opciones:

- o: Redirige la salida a un fichero `sort -o textordenado.txt texto.txt` (Sería lo mismo que hacer `sort texto.txt > textordenado.txt`)
- c: Mira si el texto está ordenado y te avisa si no.
- r: Ordena al revés
- k: Ordena por la columna que queramos

Strings:

Busca caracteres imprimibles en un archivo. Ej: `strings archivo`

Uniq:

Elimina las líneas repetidas en un archiv y lo muestra. (Solo funciona si las líneas están juntas. A veces conviene hacer primero un sort. Ejemplo: `sort data.txt | uniq -u`)

Uso: `uniq archivo.txt`

-D: Imprime las líneas repetidas.

-u: Imprime las líneas no repetidas. (unique)

Tr:

Sirve para modificar unos caracteres por otros en un texto. Ejemplo: `tr '[a-z]' '[n-za-m]'`

Con -d podemos quitar caracteres: `tr -d \n` me quita los saltos de línea

Wc:

Sirve para contar las palabras en un texto (por defecto lo hace con -w).

-l: Cuenta las líneas

-c: Cuenta los caracteres

Xxd:

Pasa un archivo de Ascii a hexadecimal

Otros comandos son: `fmt`, `fgrep`, `egrep`

Alias

Comando: `alias comandocorto="comandolargo"`

Atajos

Ctrl+A: Inicio de comando

Ctrl+E: Fin del comando

Ctrl+C: Detiene el comando de forma segura

Ctrl+R: Histórico comandos con lo que indroduzcas

Ctrl+Flechas: Salta por las palabras

Ctrl+U: Quita lo que hay antes del cursor

Ctrl+K: Quita lo que hay después del cursor

Ctrl+Z: Fuerza la detención del comando (lo detiene y lo manda a segundo plano)

!& en línea de comandos: Referencia al último argumento.

Directorios y archivos

Compresión:

Tenemos varias herramientas de compresión y empaquetamiento (lo mismo que archivado):

Zip: Comprime y empaqueta archivos

Comprimir: `zip archivo`

Descomprimir: `unzip archivo`

Tar: Es una herramienta de empaquetamiento, no comprime en sí, conservamos el metadata y también funciona mejor para directorios recursivos, ya que de por sí funciona recursivamente.

Depende el path relativo o absoluto, si pongo el absoluto se descomprime siempre en el path donde estaba. Con `-r` añadido algún archivo y con `-u` hacemos un tar incremental. `-t` para ver el contenido;

`-f` para el nombre del archivo (siempre al final), `-v` verbose (más detallado), `-x` para extraer.

Con `-C` (Mayus.) cambiamos donde se extrae. **Ejs:**

Empaquetar: `tar -cvf archivo`

Si queremos conservar el archivo y darle otro nombre al paquete o empaquetar varios:

`tar -cvf paquete.tar file1 file2`

Desempaquetar: `tar -xvf archivo`

Se suele usar conjuntamente con `gzip` o `bzip2` para comprimirlo adeás de empaquetarlo, para esto añadimos `-z` (`gzip`) o `-j` (`bzip2`) **Ejs:**

Empaquetar y comprimir: `tar -cvzf archivo`

Desempaquetar y descomprimir: `tar -xvzf archivo.tzr.gz`

Gzip (extensión .gz): Comprime pero no empaqueta archivos.

Comprimir: `gzip archivo`

Descomprimir: `gzip -d archivo.gz` o `gunzip archivo.gz`

Bzip2 (extensión .bz2):

Comprimir: `bzip2 archivo`

Descomprimir: `bzip2 -d archivo.bz2` o `bunzip2 archivo.bz2`

Star: Funciona como tar pero soporta SELINUX.

Comprimir: `star -c -f=file.star archivo` (Probar sin archivo y con la c y f juntas, por si aca) También podemos poner directorio

Descomprimir: `star -x -f=file.star`

La opción `-H=exustar` no pierde las ACL de archivo. Para guardar los atributos del archivo hacemos `-xattr`

Creado y borrado:

Crear: `mkdir directorio`, con `mkdir direct(1,2,3)` creamos `direct1` `direct2` y `direct3`

`touch file.txt`, creamos `file.txt` vacía.

Con `-p` creamos también los directorios padres necesarios.

Con `rmdir` borramos directorios vacíos.

`Cd` a secas te lleva a birgulilla.

Con `rm` borramos archivos. Con `-f` forzamos y hacemos que no pregunte, con `-r` borramos directorios también.

File:

Sirve para ver el tipo de un fichero: `file fichero.txt`

Find:

Find busca siempre de forma recursiva desde donde busques

Comando para buscar: `find <dondebuscamos> <parámetros>`

`-type`: Por tipo: `f` fichero `d` directorio `l` enlace

`-name`: Por nombre fichero, `-iname` no distingue minúsculas y mayúsculas

`-size`: Por tamaño: Bytes `c`, Kilobytes `k`, Megabytes `M`, Gigabytes `G`. Ejemplo: `+500M`, `-500M` o `500M`.

`-cmin`: Ve según los hayamos creado `-5`, hace menos de 5 min o `+5` hace más. La `c` es creación, `m` modificación y `a` acceso. Podemos cambiar `min` por `time` y entonces lo vemos por días.

`-user` `-group` `-perm`: Busca por propietario. Usuario, grupo y derechos de acceso. Por ejemplo para buscar archivos con `setUID` haríamos `-perm -u+s`

`-empty`: Busca archivos vacíos.

`!`: Busca al revés

-exec: Te ejecuta lo que tu le pongas. Si ha encontrado la cadena de texto entre {} (Normalmente se deja así para que cuente todos los resultados del find)

Luego al final comentamos el ';' para que no lo vea como final del comando find sino del comando que se ejecuta con el exec, se pueden añadir varios seguidos.

Por ejemplo un exec de grep -q primero para filtrar lo que contenga x y luego un rm

Ej:

```
find / -type f -mtime -80 -exec cp {} /copia ';' ;
```

Con esto copiamos los ficheros modificados hace menos de 80 días al directorio /copia con un solo comando.

Listado:

Usamos el comando: `ls`

Para mostrar como una lista: `ls -l`

Para ordenar archivos por tamaño usamos

-a: Para mostrar archivos ocultos: `ls -la`

-r: Para invertir el orden usamos la: `ls -lr`

-t: Para ordenar por hora y fecha, se usa con la -r para ver los archivos más nuevos abajo:

```
ls -lrt
```

-i: Para mostrar los inodos de los archivos o subdirectorios: `ls -lia`

-s: Para mostrar el tamaño en kb de los archivos

-R: Para listar también los directorios recursivos: `ls -lR`

-h: Para ver de manera más clara el tamaño de los archivos: `ls -lh`

--author: Para saber quién creó el archivo: `ls --author -l`

-n: Para mostrar el suid y el sgid: `ls -n`

`ls -Z`: Vemos la conf del archivo de SELINUX, vemos user, role y context (lo importante, para ver los posibles contexts /etc/selinux/targeted/contexts). También existen `id -Z` y `ps -Zaux`

Crontab:

Gracias al crontab podemos automatizar tareas en linux.

El crontab del sistema se encuentra en /etc/crontab y el de los usuarios en /var/spool/cron

Con `crontab -e` creamos un crontab y con `crontab -l` vemos los cronjobs existentes.

El servicio se llama crond.

El formato al añadir tareas es:

*** * * * *** El asterico significa se ejecuta en todos + el comando que queremos

1: Minutos 0-59 > */5 = Cada 5 minutos

2: Horas 0-23 > 3-6 = Entre las 3 y las 6

3: Día 1-31 >

4: Mes 1-12 > */2 = En los meses que son divisibles por dos: Febrero...

5: Nombre del día 0-6 0=Domingo > 0,6 = Sábado y Domingo

Ejemplo: 0 0 * * 0 echo "Viva Asturias" >> /home/mcm/Asturias.txt = A medianoche los domingos añade Viva Asturias al fichero Asturias.txt

Para programar scripts pero que nos den igual la hora, simplemente los metemos en /etc/cron.daily hourly... (Eso se llama anacron)

Cron.allow y cron.deny (No pueden existir los dos ficheros a la vez) Metemos en uno o en otro los que queramos permitir/denegar, los demás harán lo contrario del fichero.

At: Se usa para tareas que tienen que ser ejecutadas solo una vez.

Ej: at 20:00; at now + 10 minutes

Luego pones los comandos y Ctrl+D para salir, con atq vemos las tareas que se van a ejecutar

Espacio en disco

Uso del espacio de disco: df

Para que no muestre el espacio en kbytes y sea más amigable usamos el parámetro más usado: df -h (Importante la h en minúscula, sino te redondea)

Para ver cuanto espacio de disco ocupa un directorio: du Descargas

Aquí el -h sirve para lo mismo: du -h Descargas

Para que no nos aparezcan los subdirectorios usamos -s: du -hs Descargas

También podemos mostrar el tamaño de cada archivo: du -a

La podemos canalizar en orden para ver una lista ordenada de archivos, el parámetro «-n» le dice al comando sort que considere la primera columna de números en la salida de du como una cadena numérica: du -a Descargas | sort -n

Enlaces

Son útiles para crear accesos directos y poder crear varias ubicaciones para las carpetas.

Para crear el enlace usamos ln, para que se borre el archivo necesitaremos borrar todos los enlaces.

Se puede mirar el número de enlaces que tiene con stat link.txt

Con `-s` le indicamos que es simbólico. Se rompen si se borra el archivo original.

Ej: `ln -s archexiste nombrenlacesim`

Podemos comprobarlo haciendo `ls -l` en la carpeta donde hemos creado el enlace.

Para eliminar los enlaces existentes usamos `unlink enlace`, aunque también podemos hacerlo con `rm enlace`.

Gestión de usuarios

Creamos usuarios con `adduser` (forma corta) o `useradd` (forma larga). Los borramos con `userdel`. (Para grupo cambiamos por `group`)

Con `adduser -s` cambiamos la shell, la estándar es `/bin/bash`, si queremos ponerle non-interactive shell le ponemos `/sbin/nologin`.

También podemos ponerle un UID específico con `adduser -u`

Los usuarios los modificamos con `usermod` y los grupos con `modgroup`.

Con `deluser --remove-all-files user`.

Con `su` cambiamos de usuario. Ejemplo: `su migue`

`-l`: Se usa para cambiar completamente el entorno y usar el del usuario al que cambiamos.

GRUB

El GRUB (Grand Unified Boot Loader) es el cargador de arranque del sistema incluido en Red Hat. Un pequeño trozo de código que permite cargar el SO que desees al encender la máquina. Auqn

Aunque solo tengamos un SO RedHat en el GRUB encontraremos los kernels más recientes para arrancar (si hay fallos al hacer una update podríamos arrancar desde el antiguo) y el modo de rescate. Para entrar en cada opción usamos `e`.

Config GRUB:

Para cambiar config del GRUB:

`grub2-` (Tabulas y ves los comandos)

`grub2-editenv list` (Vemos las variables de entorno)

`grub2-set-default 1` (Eso la próxima vez que arranque elegiría por defecto la opción 2 del GRUB)

No se edita directamente el fichero de configuración por los posibles fallos, editamos el fichero `/etc/default/grub`

En el examen te dan ellos la variable a modificar, por ejemplo:

GRUB_TIMEOUT (Lo que tarda en elegir la opción si no tocas nada)

GRUB_TIMEOUT_STYLE=countdown

grub2-mkconfig -o /boot/grub2/grub.cfg

Con esto ponemos la configuración en el fichero, verifica que está bien.

Parámetros de config por defecto:

Muchos de estos parámetros, se configuran en el archivo /etc/login.defs

Entre ellos se encuentran la umask por defecto del server o las máx y min days por defecto de las contraseñas entre otros.

Para cambiar la logintud mínima de las contraseñas en cambio, se hace desde el fichero /etc/security/pwquality.conf

Reseteo RedHat 8/9:

En el GRUB, entramos a la opción de rescue con e, al final de la línea de linux (con Ctrl E) borramos quiet y rhgb y ponemos rd.break enforcing=0

(esto lo que hace es pone el modo permisivo del SELinux, pero solo en este arranque así que no hay que preocuparse si luego pide otro modo en el examen)

Tip: Cuidado que te cambia el idioma del teclado al inglés.

Hacemos Ctrl X para aceptar.

Una vez en la consola hacemos:

```
mount -o remount,rw /sysroot
```

```
chroot /sysroot
```

Cambiamos la pass.

Si no hemos puesto el enforcing=0 luego hacemos: touch /.autorelabel

Luego hacemos mount -o remount,ro /

Hacemos exit dos veces

Cuando arranque, si lo hemos hecho con el enforcing=0 para poner correctala config de SELINUX del fichero que hemos tocado al cambiar la pass hacemos: restorecon /etc/shadow

Si queremos dejar la config de SELINUX como estaba: setenforce 1

LDAP

Protocolo Ligero de acceso a directorios, se utiliza para la autenticación de servicios de directorio. Lo utilizan el correo electrónico y otros programas para buscar información en un servidor.

Comandos más usados:

ldapsearch: Para buscar

ldapadd: Para añadir usuarios

ldapdelete: Para borrar usuarios

Firewall y puertos

Con el firewall restringimos acceso a la red. Viene por defecto deshabilitado.

Firewalld:

Usaremos firewalld que es el daemon de firewall de Red Hat (por defecto viene deshabilitado), para monitorizar el tráfico nuestra red.

Funciona por zonas, donde añades los servicios que permitiremos y la interfaz a la que afecta.

Para ver las zonas: `firewall-cmd --get-zones`

Para ver la configuración de una zona en específico: `firewall-cmd --zone work --list-all`

Para crear una nueva zona:

```
firewall-cmd --new-zone nzona --permanent
```

(Si creas reglas y no le pones el permanent se limpian con el reinicio)

Después de cada cambio para que tenga efecto hay que hacer: `firewall-cmd --reload`

Para añadir un servicio a una zona:

```
firewall-cmd --zone nzona --add-service=nserv --permanent
```

Ej: `firewall-cmd --zone servers --add-service=ssh --permanent`

Para añadir la interfaz a la zona:

```
firewall-cmd --change-interface=ninter --zone=nzona --permanent
```

Para ver las zonas activas:

```
firewall-cmd --get-active-zones
```

Para poner una zona por defecto:

```
firewall-cmd --set-default-zone=nzona
```

Para ver todos los servicios a los que puede afectar el firewall:

```
firewall-cmd --get-services
```

Para añadir un servicio (Lo añade a la zona por defecto si no pones zone):

```
firewall-cmd --add-service=http --permanent --zone=nzona
```

Para añadir un puerto: `firewall-cmd --add-port=8080/tcp --permanent`

Para eliminar un servicio o puerto:

```
firewall-cmd --remove-service=http --permament
```

```
firewall-cmd --remove-port=8080/tcp --permanent
```

UFW:

Es la otra opción a firewalld en Linux.

Ficheros de config: `/etc/default/ufw`

Políticas por defecto (Opciones ACCEPT, DROP y REJECT):

INPUT: Los paquetes de entrada

OUTPUT: Paquetes de salida

FORWARD: Si el servidor actúa como router

Para cambiarlas: `ufw default allow incoming` (o `deny` | o `outgoing`)

Ver versión: `dpkg -s ufw | grep -i "version"` o con `ufw version`

Lo podemos actualizar con: `apt install ufw`

Para habilitar ssh: `ufw allow ssh/tcp`

Para habilitar el firewall (si estás por ssh primero haz lo de arriba):

`ufw enable` (disable deshabilitamos)

Para ver el status: `ufw status verbose` (para ver más info)

Para permitir la entrada solo por un puerto: `ufw allow 80`

Para permitir la entrada solo a un host:

`ufw allow from 192.168.1.3 to any port 8000`

`ufw allow from 192.168.1.3 to any port 8000:8200 proto tcp` (Rango de puertos)

Para eliminar reglas:

Por número [`ufw status numbered` (para saberlo)]: `ufw delete 7`

Abrir: `ufw allow NPUERTO`

Cerrar: `ufw deny NPUERTO`

Para rango de puertos: `ufw deny NPUERTOINI:NPUERTOFIN/tcp` (si es necesario ponemos protocolo)

Puertos comunes número y para qué se usan:

[List of TCP and UDP port numbers - Wikipedia](#)

Permisos

Chmod/own:

Este comando lo usamos para modificar los permisos de un archivo.

Para modificar el propietario usamos `chown`.

Ej: `chmod 777 archivo ; chown migue utilidades_linux`

Posición números: Usuario, grupo, otros.

Valores: 4 read, 2 write 1 ejecutar(x). Los sumamos para el número final.

-R para recursivo (para cambiar carpetas)

Hay tres permisos especiales (Si quieres poner varios los sumas como en los normales):

Sticky Bit (T en others): Solo pueden ser borrados o renombrados por el propietario del archivo, del directorio o el usuario root aunque el resto tengan permisos de escritura.

Añadiendo un 1 en el octal antes **Ej:** `chmod 1777 archivo`

Sgid (S en group): Llamado también setGID. El usuario que ejecute el fichero tendrá durante la ejecución los mismos permisos que el grupo propietario (Mismo que `suid` pero para grupo)

Añadiendo un 2 en el octal antes **Ej:** `chmod 2777 archivo`

Suid (S en owner): Llamado también setUID, sirve para casos de ejecuciones de ficheros que necesitan permisos de otro fichero. El usuario que ejecute el fichero (que necesita permisos) tendrá durante la ejecución los mismos permisos que el usuario propietario (los dos ficheros tienen el mismo propietario en estos casos siempre). Sirve por ejemplo para obligar a editar un fichero con un ejecutable.

Añadiendo un 4 en el octal antes **Ej:** `chmod 4777 archivo`

Para poder añadir a varios archivos algún permiso para todos los usuarios sin tener que mirar lo que tiene podemos usar `chmod +r,+w o +x archivo`

-R recursivo.

ACL (Access Control List):

Sirve para necesitamos permisos más específicos que los permisos normales. Para que se puedan usar el sistema de archivos tiene que estar montado con ACL, existen de grupo y de usuario.

`getfacl *` : Vemos las ACL de cada fichero, si tuvieran aparecerían debajo de los permisos de toda la vida.

`setfacl`: Ponemos la ACL, parámetros `-m g:sales:rx /dir | -m u:linda:rwX /data`.

Para quitarlos: `setfacl -x`.

Para hacer un backup de las ACL: `getfacl -R /directorio > archivo.acls`.

Para restaurarlo: `setfacl -restore=file.acl`

Umask:

Son los permisos que se ponen por defecto al crear un archivo. El formato es 777 menos la umask son los privilegios con los que se crean los archivos. Se fija en `/etc/profile`

El comando `umask` devuelve la máscara activa en esa sesión. Con `umask NUM` cambiamos la máscara activa. **Ej:** `umask 0077` (Así se crearían los archivos con permisos 700)

Atributos de fichero:

Están por encima de los privilegios:

A: Deshabilita el los metadatos cada vez que se accede a un fichero.

a: Se puede en el final del fichero pero no se puede borrar. Por ejemplo ficheros de logs.

c: El archivo se escribe comprimido.

D: Se escribe inmediatamente sin caché.

d: No añade el archivo en copias de seguridad con la utilidad `dump`.

I: Genera un índice

i: Hace el archivo inmutable

j: Va al journal de los ext3, por si estás escribiendo mientras se cae el sistema.

s: Cuando escribes en el archivo escribes con 0s.

Se cambian con `chattr`, se miran con `lsattr` **Ej:** `chattr +d sda1`

No todos los sistemas de archivos soportan todos los atributos, ante la duda mirarlo en man.

Procesos

Hay tres tipos principales de procesos: Shell Jobs (Tareas de la shell), Daemons (Procesos de Servicios) y Kernel Threads (Procesos del Kernel, entre corchetes en el `ps aux`).

Ps:

Este comando lo usamos para ver los procesos del sistema.

La información que vemos es:

PID - Número del proceso.

TIME - Tiempo que lleva corriendo el proceso.

TTY - Nombre de la terminal que controla el proceso.

CMD - Nombre del comando con el que se inició.

ps aux: Opción común donde le decimos que lo muestre más detallado.

ps -ef: Opción donde me da todos los procesos del sistema, ya que el ps normal solo da los hijos (e para que muestre todos y f para que diga la shell padre). Se suele usar con grep para buscar algo concreto.

Para matar los procesos hacemos `kill -9 PID`, con `-1` les hacemos reload y sin número (equivalente a `-15`) lo paramos ordenadamente. Con `pkill name` los matamos por nombre (si hay varios matamos a todos)

Para correr un proceso en segundo plano le ponemos `&` al final. Vemos los que tenemos en segundo plano con `jobs`. Lo enviamos a primer plano con `fg` (el último), sino hacer `fg Numero` (El número que aparece en el comando `jobs`).

Con `bg` podemos hacer que siga corriendo en segundo plano un proceso parado con `Ctrl+Z`

Wait:

Espera a que un proceso se termine de ejecutar (Útil en scripts)

Ej: `wait 32`

Con `nohup`, se lanza un proceso en local, sigue corriendo aunque se caiga la sesión, se suele lanzar con `&` para que corra en segundo plano. `Screen` hace lo mismo pero en una sesión ssh (yum install -y screen)

Con `screen -r` me vuelvo a reconectar.

Cada proceso tiene una prioridad (del 0 máxima al 39 mínima). Por defecto arrancan con 20 de prioridad. Se la cambiamos con `nice` y `renice`. Ej: `renice -n -20 -p PID` (Le quitamos 20 a la prioridad, es decir más prioritario) `nice -n 5 procesonuevo` (Se lanza con 25 de prioridad).

Con `top` vemos los procesos en una ventana de texto y nos muestra bastante info. Con `h` mostramos la ayuda para ver opciones para ordenarlos dentro de este.

Ahí podemos ver los zombies, que consumen muchos recursos

Otros: `htop`, `atop` `isof`, `nmon`, `iostat`, `sar`, `vmstat`.

`lsdf idproceso` - me dice los archivos que está usando ese proceso

Servicios

En los Red-Hat modernos se usa Systemctl, que funciona por units (service, sockets...), crea un fichero de configuración y cuando se instala el programa nosotros ya podemos gestionarlo. Para info:

```
systemctl -t help
```

Comandos: `systemctl (start,status,stop,restart,reload,enable o disable) nombreserv`

Con reload lo recargamos pero no lo reiniciamos del todo, con enable hacemos que se inicie en cada arranque.

Para listar los servicios activos podemos usar, si añadimos `--all` listamos también los inactivos:

```
systemctl -type=service
```

Para listar los servi

Por si no tenemos systemctl (otras versiones), probar:

```
service nombreserv (start,stop,restart)
```

Para editar un servicio directamente con systemctl se puede usar, te valida directamente los cambios:

```
systemctl edit nombreserv
```

```
export SYSTEMD_EDITOR="/bin/vim" (Esto para que lo edite con vim)
```

Con `systemctl mask servicio` hace el servicio inválido hasta que lo cambiamos

(lo redirige al dev/null)

(Sirve por ejemplo si tenemos servicios que se inician al arrancar no tener que cambia la config si queremos durante x tenerlo que no funcione)

Para quitarlo mismo comando pero con `unmask`

Red y parámetros del sistema

SSH:

Generación Claves pública y privada:

`ssh-keygen`: Crea en la carpeta `ssh` el `id_rsa` (clave privada del servidor) y el `id_rsa.pub` (clave pública)

Luego copiamos la `id_rsa.pub` al `authorized_keys` (lo creamos con permisos 600 si no está creado) para que me reconozca la clave

Para copiar la clave pública a otro serv linux usamos `ssh-copy-id user@maquina` (verificamos que se copia en la carpeta `.ssh` en el fichero `authorized keys`)

En el ssh te sirve para conectarte desde la maq donde haces el keygen a donde llevas la pública.

Para Windows es diferente (repasar(creo que no entra en el RHCSA))

Resto:

El archivo de configuración está en `.ssh/config`

Los hosts conocidos están en `.ssh/known_hosts`, si una máquina reutiliza una IP borrar la línea del host de antes para conectar.

Para conectarse a una máquina desde otra: `ssh usuario@IP(o nombre maq)`

`ssh-copy-id usuario@IP`: Con esto copiamos la clave para que en la próxima no nos pida contraseña

`ssh -t usuario@IP comando`: Con esto ejecutamos comandos directamente en la máquina

`ssh -D 9999 usuario@IP`: Crea un proxy

`ssh -X usuario@IP`: Me conecto y si por ejemplo ejecuto firefox, lo muestra en mi máquina.

`ssh -L Puertomimaq:IpMaq2:Puertomaq2 usuario@IPmaq` :Pegas 2 saltos de golpe si la segunda tiene firewall

`ssh -R 2020:localhost:22 usuario@externo` :Haces un ssh “al revés”.

Luego en tu maq haces `ssh root@localhost -p2020` y te conectas a la otra máquina

SCP sirve para pasar archivos y usa ssh (también existe rsync).

Para copiar archivos de tu máquina a otra:

`scp file.txt usuario@externo:/root/`

Para copiar archivos desde otra máquina:

`scp usuario@externo:/root/file.txt /tmp`

`-r`: Copia recursivamente

FTP:

Se usa para transferir archivos de una máquina a otra: ftp

`-p`: Puerto

Uname:

Comando que sirve para ver parámetros del sistema. Opciones:

`-n`: Nombre del host

`-r`: Versión kernel

`-m`: Arquitectura de hardware

Traceroute:

Sirve para ver todos los pasos en una conexión de red.

Ejemplo: `traceroute tryhackme.com`

Para cambiar interfaz usamos `-i`

Ping:

`-q`: Lo hace quiet

`-c 2`: Lo hace 2 veces

Otros:

`Ifconfig`: Ve config de interfaces (Ubuntu).

`ip addr`: Ve IP e interfaces (Ubuntu).

`ip a`: Ve IP e interfaces.

`ip r`: Ve IP e interfaces.

`ip -s link show`: Mira estadísticas de nuestra conexión de red.

`Hostname`: Ve nombre de host y dominio, `-i` y `-I`.

`uptime`: Vemos el tiempo que lleva encendida la máquina.

`w`: Lo mismo pero da algo más de info

`dmesg`: Muestra eventos del kernel (por si conectas dispositivos o algo), para que quede esperando `-follow`

`strace`: te muestra eventos de sistema (por ejemplo comandos)

`mtr`: un traceroute en vivo

`lshw`: veo las propiedades del hardware

`lspci`: Me da info de los dispositivos conectados.

`lscpu`: Me da info de la CPU.

Directorios y archivos importantes

/boot:

Carpeta del cargador de arranque, necesaria para iniciar la máquina.

/dev:

Es un sistema de ficheros virtual. Contiene archivos de los dispositivos conectados al sistema, por ejemplo los discos duros conectados al sistema

Allí nos encontramos archivos como:

/dev/null: Un sistema de archivos que no existe, lo que entra ahí se pierde para siempre. Se usa para redirigir salidas de comandos que no necesitamos.

/etc:

Archivos de configuración de programas propios del SO y de programas instalados por el usuario (Todo lo que queramos de configuración):

Allí nos encontramos archivos como:

/etc/redhat-release: En una máquina red hat, me dice la versión.

/etc/hosts: Mapea los nombres de los equipos con las correspondientes IPS.

/etc/hostname: Contiene el nombre de la máquina.

/etc/passwd: Contiene información de los usuarios (nombre,UID,GID...)

/etc/crontab: Sirve para automatizar tareas en el sistema.

/etc/resolv.conf: Sirve para resolver los nombres DNS.

/etc/motd: El mensaje que te sale después de hacer el login.

/etc/issue: Lo mismo pero antes.

/etc/sudoers: Configuración de permisos de sudo, con `visudo` podemos editarlo directamente.

/home:

Donde los usuarios almacenan sus datos y archivos de configuración. A veces se monta con `noexec` o `nodev` (Para que no se pueda montar dispositivos en /home) para más seguridad (Va por encima de los permisos).

/media o /mnt:

Son físicos y están vacíos, se usa para montar

/proc:

Sistema de archivos virtual. Los procesos que se están ejecutando, me da info del sistema. En `/proc/irq` vemos las interrupciones del sistema en `/proc/swaps` tenemos info del swap y en `1` tenemos el `init`. Las demás herramientas de procesos leen el `/proc`

/root:

Directorio del superusuario administrador root. Si montamos / estamos montando /root, / contiene los directorios o sistemas de ficheros.

/tmp:

Carpeta donde se almacenan archivos de forma temporal. Los archivos que no se usen en 10 días (según configuremos) serán borrados automáticamente. También existe /var/tmp, que es lo mismo pero con 30 días (según configuremos) .

/usr:

Contiene software instalado, librerías compartidas, los comandos y datos de programa de solo lectura.

Allí nos encontramos archivos como:

/usr/bin: Comandos de usuario.

/usr/sbin: Comandos de administración del sistema.

/usr/local: Software local modificado.

/usr/share/doc: Páginas de documentación.

/usr/share/dicts/words: Listado de palabras en linux.

/var:

Datos variables que necesita el sistema cada vez que arranca. Se suele montar aparte de la / porque va creciendo con los logs de httpd. Allí se pueden encontrar normalmente logs, archivos que cambian dinámicamente, caché de webs...

/var/log/messages: Fichero de logs más común del SO, ahí se escriben la mayoría de los mensajes.

/var/log/dmesg: Logs del kernel.

/var/log/secure: Logs de autenticación. Los errores de autenticación se almacenan ahí.

/var/log/boot.log: Logs relacionados con el arranque.

/var/log/audit/audit.log: Contiene logs de auditoría. SELINUX escribe ahí.

/var/log/maillog: Logs relacionados con el correo (en desuso)

/var/log/samba: Logs relacionados con samba.

/var/log/httpd: Directorio que contiene los archivos escritos por Apache.

/run:

Datos de los procesos iniciados desde el último inicio. Antiguamente era /var/run

/sys:

Me da una visión del SO basada en el hardware.

Volúmenes y gestión de discos:

NAS: Network Attach Storage. Es como una serie de "discos duros externos"

El comando `dd` sirve para volcar datos de un sitio a otro (usar con cuidado):

Para guardar la tabla de particiones del disco (conviene hacerlo cuando vas a tocarlo):

```
dd if=/dev/sda of=/root/sda.part bs=1M count=1
```

Para recuperarlo hacemos el mismo comando cambiando `/dev/sda` por `/root/sda.part` y viceversa.

Para copiar en sí la tabla de particiones hacemos `cp /etc/fstab /root/fstab`

MBR (Master Blue Record): La tabla de particiones de toda la vida, se gestiona con `fdisk`. Usa particiones primarias (hasta 4) extendidas que contiene lógicas (no tienen diferencias)

El comando `fdisk` sirve para gestionar los discos y las particiones en el caso de que sea MBR.

Con `fdisk /dev/sdb` hacemos cambios al disco, desde ahí se abre un menú y podemos crear particiones.

Para crear una nueva partición le damos a la `n`. Elegimos el tipo de part (primaria o extendida) elegimos el primer sector (por defecto) y el último, por ejemplo `+3G` (G para gigas, M para megas, K para kilobytes, sino ponemos nada lo hace por sectores). Por último le ponemos el tipo (con `t`, con `h` vemos las opciones). Los más comunes son: 82 – Linux Swap; 83 – Linux File System; 8e LVM

Con `fdisk -l` podemos ver información de las particiones de los discos. Podemos ver uno en concreto `fdisk -l /dev/sdb`

GPT: Nuevo sustituto sin límite de particiones y que vale para discos muy grandes. Si la tabla de particiones es GPT, usamos `gdisk`, se usa igual a `fdisk`. En los tipos añadir 2 ceros al final.

`lsblk`: Se usa para ver las particiones.

`ls SCSI`: Vemos los discos y los puertos que corresponden.

`mount`: Vemos los dispositivos montados.

`findmnt`: No da el montaje de mi sistema de ficheros.

También tenemos la opción de `parted` en vez de `fdisk` y `gdisk`

NFS:

Un NFS es un sistema de archivos en red compartido por una o varias máquinas (ya sea el servidor una cabina u otra máquina)

Para la configuración del servidor nos vamos al fichero `/etc/exports`, donde ponemos los NFS que compartimos. También tendremos que tener instalado `nfs-utils` y activar el `nfs.server.service`, si ya estaba activo reiniciamos después de los cambios.

Luego en el cliente tenemos que montarlo:

```
mount -t nfs IPSESV:/ruta /dondese monta
```

Si queremos que sea permanente lo hacemos desde el `fstab` y `mount -a`.

También existe AutoFS, para montar automáticamente archivos en el nfs, el paquete se llama `autofs`.

AutoFS:

Después de instalar `autofs` comprobamos los IDS de los usuarios y el grupo del usuario, tienen que coincidir tanto en servidor como en cliente, sino coinciden hacemos `usermod -u ID username`. (Chequeando que no se carga ningún usuario), luego cambiamos los permisos de los archivos al nuevo id con:

```
for i in $(find / -user 1006) ; do chown IDNUEVO $i; done
```

Después de los IDS creamos el archivo de configuración del AutoNFS con `/etc/auto.master.d/home.autofs`

allí añadimos

```
/home /etc/autofs.home
```

Ahora creamos el archivo `/etc/autofs.home`:

```
* -rw IP/HOSTNAMESERVER:/home/& (lo último significa el propio home directory)
```

Ahora hacemos el `enable` y el `restart` del servicio

Ahora vemos que si creamos un archivo en el directorio del usuario se crea en las dos máquinas.

Como lo hemos configurado con el `&`, nos funciona con todos los usuarios que coincidan el ID (supongo xD)

LVM:

Normalmente en linux, al crear particiones hay un problema. Si creamos `/dev/sda1` y `/dev/sda2`, se crean con los bloques consecutivos, por lo que no podemos aumentarle el tamaño a las particiones sin borrarlas. Esto se arregla con LVM (Logical Volume Manager)

Un volumen lógico se compone de particiones lógicas asignadas a discos físicos (o a particiones) (Donde se almacenan los datos en un LVM)

El proceso que administra LVM es el `clvmd`. Si se quedan pillados los comandos de LVM lo podemos reiniciar.

Con los comandos `pvs`, `vgs` y `lvs` vemos la información de los volúmenes físicos, grupos de volúmenes y los volúmenes lógicos respectivamente, con la opción `-a` vemos más info. Con `vgdisplay` y sus homónimos hacemos igual pero con más info.

Pasos para crear o ampliar un LVM:

1- Creamos el volumen físico con LVM. **Ej:** `pvcreate /dev/sda`

Si ya hemos creado el pv y queremos aumentarlo porque hemos agrandado el disco.

Ej: `pvresize /dev/sda3`

2- Creamos un grupo de volúmenes a través de un disco físico o lo modificamos si ya está creado.

Ejs: `vgcreate vg /dev/sdb ;`

`vgcreate vg /dev/sdb /dev/sdc` (Si necesitamos dos discos) ;

`vgextend vg /dev/sdd` (Si añadimos un disco al vg)

3- Creamos un volumen lógico dentro del vg.

Ejs: `lvcreate -L 50G -n lv vg`

`lvcreate -l 100%FREE -n lv vg`

(Con el `-l` hacemos que el resto de espacio libre lo ocupe la partición. Esto viene bien si queremos hacer dos particiones de 75G en un disco de 150, ya que en verdad ocupan algo más, así creamos la última. En este caso al después de aumentar el pv no hace falta tocar el vg) ;

`lvextend -L +1G /dev/vg/lv`

(Si le queremos añadir espacio, también podemos usar el `-l 100%FREE`)

Con `-r` al extender aumentamos directamente también el sistema de ficheros:

`lvextend -L +1G /dev/vg/lv -r`

3B- Si queremos duplicar los datos de un logical volume de un disco a otro:

Ej:

`lvconvert -b --type mirror -m1 --corelog nombrevg/nombrelv /dev/sdd`

[Si queremos duplicar los datos del lv al sdd, (tendrá que ser igual o más grande que el lv)]

4- Luego crear o aumentar el sistema de ficheros (a partir de aquí se hace aunque no usemos LVM):

Nota: `/dev/mapper/nombregrupovol-nombrevollogico` es lo mismo que poner `/dev/nombregrupovol/nombrevollogico (/dev/vg/lv)`

- Podemos formatear con XFS, un tipo de sistema de ficheros, lo caracteriza que no se puede reducir. En ese caso el comando es (Mirar num inodos, por si piden inodos concretos):

Ejs:

```
mkfs.xfs /dev/vg/lv
xfs_growfs /puntomontaje
xfs_repair /dev/vg/lv (Para reparar)
xfs_info (Para ver la info)
```

- También se podría formatear con ext3 o ext4. Para eso:

Ejs:

```
mkfs.ext4 /dev/vg/lv
resize2fs /dev/vg/lv
fsck /dev/vg/lv (Para reparar)
```

- También se podría formatear con vfat, un sistema de ficheros más antiguo. Para eso:

Ejs:

```
mkfs.vfat /dev/vg/lv
fsck /dev/vg/lv (Para reparar)
```

- Si la partición fuera una partición swap:

En vez de la parte del sistema de ficheros, apagamos la swap con `swapoff -a`.
Comprobamos con `free -m`. Luego para convertir el espacio en swap hacemos `mkswap /dev/rhel/swap`. Encendemos la swap con `swapon /dev/rhel/swap` y comprobamos con `free -m`.

5- Por último solo nos quedaría montarlo si la partición es nueva, creando la carpeta si la montamos donde no existía carpeta:

Se podría hacer esto editando `/etc/fstab`, en cuyo caso editamos y luego hacemos

`mount -a` para guardar los cambios. También podemos montar por UUID (los vemos con `blkid`) o por label (como una etiqueta)

Existen varias opciones comunes al montar un disco, esto se define en la columna donde suele venir defaults:

Option	Use
auto/ noauto	The file system will [not] be mounted automatically.
acl	Adds support for file system access control lists (see Chapter 7, "Permissions Management").
user_xattr	Adds support for user-extended attributes (see Chapter 7).
ro	Mounts the file system in read-only mode.
atime / noatime	Disables or enables access time modifications.
noexec / exec	Denies or allows execution of program files from the file system.
_netdev	Use this to mount a network file system. This tells fstab to wait until the network is available before mounting this file system.

Ej fstab:

```
/dev/mapper/rhel-swap swap swap defaults 0 0
UUID="a3f9900..." /pmontaje ext4 defaults 0 0
LABEL=LogicalVol1 /pmontaje xfs defaults 0 0
```

Existen varias opciones comunes al montar un disco, esto se define en la columna donde suele venir defaults:

El penúltimo es sobre las copias de seguridad (a veces se pone 1), el último sobre los chequeos automáticos al arrancar (0 deshabilitado, 1 para el root que se chequea automáticamente y 2 para otros file system que queramos chequear al arrancar)

Los cambios que hagas con mount no son permanentes, los tienes que poner en el fstab para eso, con los nfs pasa lo mismo:

Desmontar: `umount /home/projects`

Montar: `mount nombrepart dondemontamos`

Con `-o remount` intentamos volver a montar uno ya montado anteriormente.

Para crear un label usamos `e2label`

Pasos para reducir o eliminar un LVM (cuidado con los datos):

1- Reducimos el lv (Tendrá que estar desmontado).

Ejemplo:

```
lvreduce -L 400M /dev/vg/lv
```

(El tamaño es el nuevo tamaño que queremos)

```
lvreduce --resizefs -L 5G /dev/vg/lv
```

(Si es de espacio que no estamos usando le añadimos `--resizefs` para no perder los datos)

1B- Si acabamos de duplicar los datos de un disco a otro y queremos eliminar el lv del disco antiguo (Aquí no hay que desmontar nada).

Ej:

```
vconvert -m0 vg/lv /dev/sd"X" -vvv
```

2- Reducimos el vg.

Ej:

```
vgreduce vg /dev/sdx
```

3- Si queremos eliminar el pv del disco que quitamos:

Ej:

```
pvremove /dev/sdx
```

blkid:

Para activar quotas:

```
quotaon -vu /home/backups
```

```
quotaon -vg /home/projects
```

VDO/STRATIS:

Stratis:

Aparece en Red Hat 8 y contiene varias funcionalidades: Aprovisionamiento fino, snapshots del volumen y una capa de caché entre las más conocidas. Es comparable a un LVM y funciona con pools.

Para instalarlo los paquetes se llaman `stratis-cli` y `stratisd`

Tendremos que habilitar e iniciar el servicio `stratisd`

Para crear un pool: `stratis pool create mypool /dev/sdd`

Luego habría que crear el sistema de ficheros (se crea con `xfs` directamente):

```
stratis fs create mypool volstr
```

Por último habría que montarlo, con `mount` o en el `fstab`, para esto último debemos de usar el UUID e incluir la opción `s-systemd.requires=stratisd.service`

Ej: `mount -t xfs /dev/stratis/mypool/volstr`

Para crear un snapshot:

```
stratis filesystem snapshot mypool volstr volstr_snapshot
```

Para recuperarla:

```
mount /dev/stratis/mypool/volstr_snapshot /pmontaje
```

VDO:

Significa Virtual Data Optimizer. Utiliza las características de deduplicación (reducir tamaño de los archivos) en volúmenes que contienen discos virtual de MVS. Necesita 4GiB como mínimo.

Los paquetes para instalar se llaman `vdo` y `kmod-kvdo`

Para crearlo (solo en RedHat 8) tenemos el comando:

```
vdo create -name=namevdo -device=/dev/sdd -vdoLogicalSize=1T
```

En la 9 se crea como un LV:

```
lvcreate --type vdo -L 20G --name namevdo vgvd
```

Después creamos el sistema de archivos: `mkfs.xfs -K /dev/mapper/namevdo`

(la -K para que lo haga más rápido)

```
mkfs.xfs /dev/vgvd01/namevdo
```

Para montarlo en el fstab incluiremos la opción donde se pone defaults:

```
x-systemd.requires=vdo.service,discard
```

Para ver estadísticas: `vdostats`

Redireccionamiento

Normalmente la salida de los comandos se muestra por pantalla (tanto la salida normal como los errores), para redirigirlo a un fichero (lo crea sino existe):

Redireccionar salida estándar: `ls -l > /home/salida.txt`

Para enviarlo al final de un fichero: `ls -l >> /home/salida.txt`

Para redireccionar la salida de errores usamos: `ls -l 2> /home/salida.err`

Para enviarlo al final de un fichero: `ls -l 2>> /home/salida.err`

Si queremos enviar ambas salidas: `ls -l &> /home/salida.txt`

Para enviarlo al final de un fichero: `ls -l &>> /home/salida.txt`

Con esto le pasamos un sort al primer fichero mandamos la salida al segundo (tienen que ser ficheros diferentes):

```
sort < /home/salida.txt > /home/salidaordenada.txt
```

Con tee lo vemos en la pantalla y también lo envía al fichero:

```
ls | tee salida.txt
```

Recordamos que si no queremos la salida lo podemos redireccionar a `/dev/null`

Vim:

Editor de texto de linux, para editar archivos hacemos `vim archivo`, para ir a una línea directamente `vim archivo +num`, para ir directamente a una palabra `vim archivo +/Palabra`.

Para entrar en modo insertar pulsamos `a` o `i`, para salir de él pulsamos `esc`.

Comandos:

Borramos una línea con `dd`, una letra con `dl` y una palabra con `dw`.

Para guardarlo con otro nombre escribimos `:w nombre.txt`

`G`: Final de fichero

`B`: Inicio de línea

`gg`: Inicio de fichero

`$`: Con esto vamos al final de línea

`Esc+U`: Deshacemos cambios

`Ctrl+R`: Rehacemos cambios

Para cortar texto, en modo comando hacemos `cc` para línea `cw` para palabra `cl` para letra.

Con `y` copiamos. Para hacer esto con varias líneas pulsamos en número de líneas antes.

Con `la p` pegamos texto

Para poner una línea debajo de otra en modo comando usamos:

`:g/key/norm owhat ever you want`

Para comparar dos archivos usamos `vim -d archivo1 archivo2` y con `ctrl+w w` nos movemos de un fichero a otro. Con `D+O` y `D+P` copiamos una línea de un archivo a otro.

`:r`: Con esto se le indicamos un archivo lo copiamos al archivo abierto.

También se pueden abrir archivos zip en vim, pulsando `enter` para entrar en cada uno de ellos.

Para ir a un archivo que está referenciado en otro podemos pulsar `gf` sin salir de vim.

Para correr comandos externos usamos `!:date`. No vale para hacer `:r !ls`, con lo que copiamos el `ls` de la ruta que estemos al archivo abierto.

`:help user` para abrir la ayuda de Vim

Tmux

Sirve para poder correr dos terminales a la vez.

`tmux new -s "Sesion": Nueva sesión`

`Ctrl+B` y luego `"` : Divide verticalmente (% horizontalmente)

Ctrl+B y luego x : Elimina el panel

Ctrl+B y luego espacio: Rota de horizontal a vertical

Otros

Apropos:

Me pasa comandos relacionados. Ejemplo: apropos subscription

Xargs:

Sirve para leer la salida de un comando y pasárselo a otro.

Ejemplo: `find . -type f | xargs file`

Time:

Al principio de un comando vemos lo que tarda.

Disown:

(-a opcional) Sirve para independizar tareas de la terminal. Ejemplo: `disown firefox` (Si cierro la terminal no cierra el firefox)

Criptografía:

La integridad de un archivo se puede comprobar con varios comandos:

`md5sum archivo.txt`

Curl: Me saca el contenido de una página web: (htmltotext me lo convierte a legible)

Which: Me dice si un programa está instalado o no

Swap: Es un espacio de intercambio (partición o fichero) que usa espacio en disco duro como datos temporales, como si fuera RAM, se usa cuando la RAM se agota. Con `free -m` vemos el uso de swap. Con el comando `sudo swapoff -a && sudo swapon -a` volvemos a pasar el contenido de la swap a la ram, (lo forzamos apagando y encendiendo la swap). A veces es necesario reiniciar la máquina para liberarlo del todo.

Selinux:

Para ver la configuración de SELINUX usada ahora usamos el comando `getenforce`

Para cambiarla usamos `setenforce`, que solo la cambiará temporalmente hasta el reinicio, para hacerlo permanente cambiamos el fichero `/etc/selinux`

Archivos:

Para cambiar el context lo hacemos con:

`chcon unconfined_u:object_r:tmp_t:s0 archivo.txt`

Para poner el context (y toda la conf de SELINUX) default: `restorecon archivo.txt`

Con -R lo podemos hacer a directorios y con --reference lo hacemos en referencia a otro fichero.

Servicios tiempo Sistema:

Para cambiar la hora: `timedatectl set-time 22:43:00`

Para listar las zonas horarias: `timedatectl list-timezones`

Para cambiar la zona horaria: `timedatectl set-timezone Australia/Melbourne`

Para habilitar NTP (que utiliza chrony): `timedatectl set-ntp yes`

`dnf install chrony`

`systemctl start chronyd` (También hacemos enable)

Para modificar el servidor NTP nos vamos a `/etc/chrony.conf`, allí debajo de pool podemos poner nuestro server para que saque la hora de allí:

`server1 172.20.10.1`

Tuned:

Administra perfiles para optimizar según el tipo de entorno donde estamos trabajando.

Para instalarlo: `dnf install tuned`

Lo habilitamos e iniciamos con `systemctl`.

Una vez está iniciado vemos el perfil activo con `tuned-adm active`.

Para ver los perfiles disponibles hacemos `tuned-adm list`

Para poner el o los perfiles que queramos usamos `tuned-adm profile namepro1 namepro2`

Podemos ver el recomendado con `tuned-adm recommend`

Podemos apagarlo con `tuned-adm off`

Otros:

`sudo !!`: Ejecuta el ultimo comando con `sudo`

`script`: Te graba la pantalla

`scriptreplay`: Te muestra lo grabado

Al hacer un cat a un archivo y modificarlo con tuberías no se puede redirigir al mismo archivo. Lo redirigimos a otro y redirigimos el otro al primero.

Instalación

Dependiendo de las versiones de linux se usa una utilidad u otra:

En Ubuntu y Debian: `apt install paquete`

En SuSE: `zypper install nombre_programa`

RedHat, Fedora o CenOS:

Usamos a herramienta de paquetería YUM, maneja dependencias (si un paquete depende de otro lo instala directamente), a diferencia de la que se usaba antes, RPM. DNF está reemplazando poco a poco a YUM. Module streams: Conjunto de paquetes para instalaciones grandes.

```
yum/dnf install paquete
```

Para instalar un paquete sin saber el nombre, pero sí el comando, podemos usar:

```
yum whatprovides */comando (Aparecerá el nombre del paquete en la primera línea, antes de la versión) dnf provides */comando
```

Si sabemos parte del nombre podemos hacer: `yum/dnf search nombre`

Para ver los programas instalados hacemos: `yum history`

Para borrar paquetes tenemos dos opciones:

```
yum history undo ID (quita tb dependencias, si no las usa otro programa)
```

```
yum/dnf remove paquete
```

Si queremos instalar un paquete que ya tenemos directamente el rpm, podemos usar para comprobar que se instala bien:

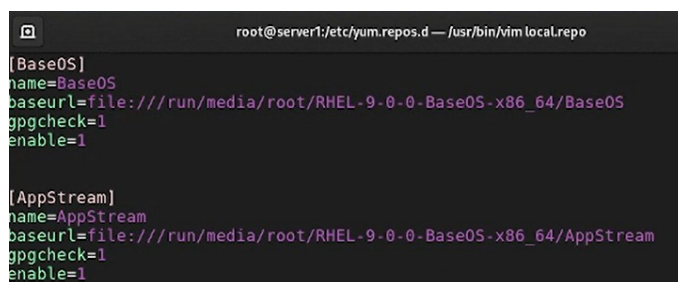
```
dnf localinstall paquete.rpm
```

Repositorios RedHat:

Con la subscripción a RedHat pilla los repos directamente, pero sino tenemos que montarlos, por ejemplo en el examen. Yum mira los ficheros del directorio `/etc/yum.repos.d`

El fichero tiene las secciones `baseurl` (la URL de la localización de los repositorios) `name` (nombre del repositorio que quiero usar) y `[label]` (si el fichero `.repo` contiene varios repositorios, los separa en secciones), luego es recomendable que tenga las secciones `enable` (para mantenerlo habilitado) y el `gpgcheck` (ambas a 1)

Ej:



```
root@server1:/etc/yum.repos.d — /usr/bin/vim local.repo
[BaseOS]
name=BaseOS
baseurl=file:///run/media/root/RHEL-9-0-0-BaseOS-x86_64/BaseOS
gpgcheck=1
enable=1

[AppStream]
name=AppStream
baseurl=file:///run/media/root/RHEL-9-0-0-BaseOS-x86_64/AppStream
gpgcheck=1
enable=1
```

En sí los contenidos se guardan en la carpeta `/run/media/root/RHEL...`

Si usamos la opción recomendable de `gpgcheck` hay que importar la clave:

```
rpm --import /run/media/root/RHEL-9.../RPM-GPG-KEY-redhat-release
```


Para ver los repositorios incluyendo los deshabilitados:

```
yum/dnf repolist all
```

Para ver los habilitados:

```
yum/dnf repoinfo
```

Bash Scripting

Los scripts de bash tienen de extensión .sh

El script siempre lo empezamos con `#!/bin/bash`. Acordarse de hacer ejecutable el archivo. Lo ejecutamos con `bash hola.sh` o con `./hola.sh`. Las variables van sin espacios.

Recordatorio sentencias:

IF: significa si, si x entonces y, recordar el then:

Ejemplo:

```
if [ "$arg1" = "$arg2" ] && [ "$arg1" != "$arg3" ]; then
```

```
    echo "Two of the provided args are equal."
```

```
    exit 3
```

```
elif [ $arg1 = $arg2 ] && [ $arg1 = $arg3 ]; then
```

```
    echo "All of the specified args are equal"
```

```
    exit 0
```

```
else
```

```
    echo "All of the specified args are different"
```

```
    exit 4
```

```
fi
```

```
while,do,done; until (contrario while)
```

for, in: Va recorriendo hasta que salga. También usa do y done. Ejemplo: `for cups in {1..10};`

`$(cat españa.txt)` : Le indicas con el dolar que es un comando, lo podemos añadir a un for. (Con dos paréntesis es con función matemática)

Ejemplo: `for x in $(cat españa.txt)`

read: Pide una variable y la lee

-p: Pide enter para continuar un while

Break y continue: Se usan en loops, break sale del bucle y continue pasa a la siguiente vuelta y se olvida de esa.

sleep 2: Se usa para esperar x segundos (En este caso 2)

`$()` : Para mates dentro

```
case $class in
    1)
        type="Samurai"
        hp=10
        attack=20
        ;;
```

Variables de entorno:

`$RANDOM` : Elige un número aleatorio entre 0 y 32747 (para poner otro % 2)

`$USER` : Devuelve el usuario

`$PWD` : Devuelve el directorio de trabajo

`$HOSTNAME` : Devuelve el nombre de la máquina

`$SHELL`: Devuelve la shell que estás usando

Crear variable de entorno:

`twitter="Elon Musk"` , la exportamos con `export twitter`

Para hacerla permanente editamos `.bashrc`: poniendo `export twitter="Elon Musk"`

Crear variable local:

`local result = ""`

añadimos lo que queramos, luego la leemos con `read result`

Matemáticas:

En bash, (no solo para scripts), para hacer operaciones matemáticas usamos dos paréntesis. Si queremos mostrarlo tenemos que hacerle un `echo $((2+2))`

`*` : Multiplica `/` : Divide `%` : Da el resto de una división

Input y Output:

0: Estándar input 1: Estándar Output 2: Estándar error

`exit 0` o `1`: Sale del script con código de éxito (2 para error)

Para pasar los errores a salida normal: `2>&1`

Opciones en if:

`-gt`: Comprueba si un número es más grande que x (lt al revés)

`-f`: Comprueba si el argumnto mandado es un archivo que existe y es un fichero.

`-d`: Lo mismo con un directorio.

`-h`: Lo mismo con un enlace simbólico.

-s: Comprueba si el archivo existe y no está vacío.

-r: Comprueba si el archivo es legible.

-w: Comprueba si el archivo tiene permisos de escritura.

-x: Comprueba si el archivo es ejecutable.

-nt: Comprueba que archivo es más nuevo, (-ot lo hace a la inversa).

-z: Comprueba que la variable esté vacía.

<: Comprueba que el primer argumento está alfabéticamente antes que el segundo (> al revés).

!=: Comprueba que no sean iguales los dos argumentos.

|| : Es un OR en un if

&& : Es un AND en un if

Otras:

Opciones en while:

-le: Menor que algo. Ejemplo: while [[\$x -le 100]]

```
cat /etc/passwd | while read line; do
```

```
    echo line
```

```
done
```

Me mostraría todas las líneas del fichero.

```
while read line; do
```

```
    echo line
```

```
done < /etc/passwd
```

Esto es lo mismo.

Let contador+=1: Suma 1 por cada paso a contador

Echo -e: Habilita la interpretación de caracteres tras \:

\a : Alerta

\e \E : Escape

\n : Nueva línea

\t : Tabula horizontalmente

\v : Tabula verticalmente

\$1: Hace referencia al primer argumento pasado al script

echo -n: No me salta de línea entre ese echo y el siguiente

una variable que va sumando en un bucle, si la lees fuera te vale lo que valía antes del bucle, para eso si queremos ver el valor final lo podemos meter a un fichero con >

Funciones:

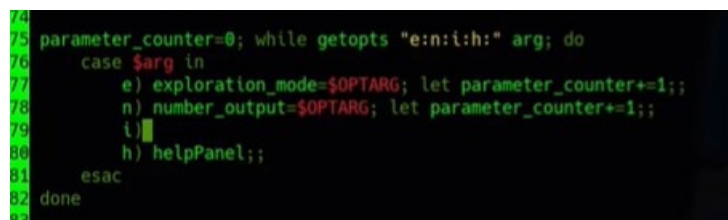
Su sintaxis es así:

```
function hola(){  
    echo "Hola"  
}
```

Luego tenemos que llamarlas, si le queremos meter una variable añadiéndola. Ejemplo:

hola

hola "\$1"



```
74 parameter_counter=0; while getopts "e:n:i:h:" arg; do  
75     case $arg in  
76         e) exploration_mode=$OPTARG; let parameter_counter+=1;;  
77         n) number_output=$OPTARG; let parameter_counter+=1;;  
78         i) ;;  
79         h) helpPanel;;  
80     esac  
81 done  
82  
83
```

OPTARG sirve para poder pasarle argumentos además de los parámetros. Ejemplo: ./miscrypt -a parametro1

(Los dos puntos al final, es que si no pones nada, te lleva a la última opción)

Con el while getopt se pueden elegir varios argumentos

```
while getopts "a:b:c:" arg; do
```

```
case $arg in
```

```
    a) funcion1=$OPTARG;
```

```
    b) funcion2=$OPTARG;
```

```
    c) funcion3=$OPTARG;
```

```
esac
```

```
done
```

tput cnorm: Para poner el cursor (tput civis para quitarlo)

; Es como si cambias de línea

set -f. Quita el globbing (*? y esos caracteres), nos vale para scripts que nos den fallo si ponemos asteriscos

Al llamar a funciones, podemos llamarlas con una variable al lado y esa será la variable \$1 de esa función¿?

Child Shells:

Una child shell se crea si pones bash en la terminal, no hereda ni alias ni variables. Se puede comprobar que la hemos creado haciendo un ps. Si hacemos Ctrl+E volvemos a la Shell padre.

Subshells:

Con echo \$BASH_SUBSHELL comprobamos si estamos dentro de una subshell o no.

Si lo hacemos entre paréntesis estamos creando una subshell. Las subshells sirven para poder juntar comandos.

Ejemplo: Si hacemos ls -l /tmp; date | wc -l en una shell normal, el wc -l nos va a contar las líneas del segundo comando.

Si hacemos el mismo comando pero los dos primeros en una subshell , el wc -l nos lo hace a los dos comandos (Parecido al paréntesis en mates)

Si yo agrego delante \$, es decir ejecuto \$(date -u), coje la salida de date -u y lo ejecuta como si fuera un comando. Sirve por si quiero hacer echo de varios comandos juntos.

#Colours

greenColour="\e[0;32m\033[1m"

endColour="\033[0m\e[0m"

redColour="\e[0;31m\033[1m"

blueColour="\e[0;34m\033[1m"

yellowColour="\e[0;33m\033[1m"

purpleColour="\e[0;35m\033[1m"

turquoiseColour="\e[0;36m\033[1m"

grayColour="\e[0;37m\033[1m"

Opciones echo útiles bash scripting:

-e: Habilita la interpretación de caracteres tras \:

\a : Alerta

\e \E: Escape

\n: nueva linea

\t: Tabula horizontalmente

\v: Tabula verticalmente

También sirve esto para el tr

```
sudo service ntp stop  
sudo ntpd -gq  
sudo service ntp start
```