**Curso** **Linux**

**composicion del arbol de directorios**

la raiz

/

**Boot/**

group y lilo son gestores de arranque, asocian el arranque del sistema operativo y la version de software que tiene que arrancar.

Kernel e imagenes se guardan en boot

**bin/**

ficheros ejecutables del sistema (los ejecutan todos los usuarios del sistema)

**sbin/**

ficheros ejecutables solo por super usuarios (sudo)

**lib/**

librerias, conjunto de operaciones que permiten a las aplicaciones ejecutar funciones

**usr/**

la mayoria de las aplicaciones se alojan aca (datos de los programas) x lo general cuando se termina de instalar se puede poner en modo solo lectura

**/usr/local**

ficheros q se instalan de forma local y aplicaciones que se compilan localmente

**opt/**

almacena aplicaciones de software de terceros (no son open source)

**home/**

contiene el home de todos los usuarios, tiene subdirectorios q contienen los distintos usuarios de este linux.

dentro de un usuario tiene su espacio de usuarios (descargas documentos imagenes)

**/root**

el home del usuario root (super usuario sudo)

**/var**

tiene variables o datos vivos. por ejemplo los logs

**/tmp**

es el temp contiene ficheros efimeros o volatiles. se elimina automaticamente cuando la pc se reinicia (todos los usuarios tienen acceso a el) se soluciona separando el punto de montaje.

**/mnt**

alberga el montaje de dispositivos como pen drives pero este solo se usa para almacenamiento de unidades en red

**/media**

es igual que mnt pero para sistemas fisicos o locales como pen drives

**/dev**

no contiene ficheros, contiene punteros asociados a dispositivos fisicos. Tenemos la asociacion entre el contacto a nivel hardware con el sistema operativo

**/proc**

proporciona informacion del hardware de forma dinámica

**SWAP:** Es una extensión de la memoria RAM almacenada en el disco duro, guarda datos temporales necesarios que no necesitan ser buscados con tanta velocidad como los de la ram.

**SHELL**

La Shell esta compuesta por un prompt que nos da información útil como el usuario, el nombre de la maquina y el directorio donde estamos

usuario@nombre\_maquina: directorio cursor

**Varriables de entorno:**

**SHELL:** Define la Shell q estamos usando, toma el valor de /bin/bash

**USER:** Define el usuario que esta actualmente logueado en Shell.

**PATH:** Me muestra los PATH de ejecutables separados por “:” teniendo guardadas estas rutas en $PATH podemos ejecutar los programas directamente sin poner toda la ruta. PATH es case sensitive

**PWD:** Donde estoy ubicado en este momento?

**LANG:** Muestra lenguaje y codificación de caracteres

**HOME:** Almacena el Home del usuario

**COMANDOS INTERNOS:**

**Alias:** Crea un alias para un comando.

EJ: $alias rm=’rm -i’

**Exec:** ejecuta una aplicación pero cuando termina cierra la consola xq no tiene PID

**Echo**: muestra en consola lo q le ponga

**Env**: muestra las variables de entorno del sistema

**Export:** Genera una variable de entorno de tipo global

**Set:** Muestra variables de entorno q no se ven en **env** y a continuación las funciones concretas q utiliza el sistema para trabajos internos.

**Unset:** elimina una variable que hayamos creado

EJ: unset HOLA

**COMANDOS BASICOS:**

**Touch:** Crea ficheros vacios, también puede modificar la fecha y hora de creación y modificación de archivos

**Cd:** Me muevo por los directorios. Para ir para atrás se usa el .. EJ: cd ../../ me fui 2 directorios hacia atrás.

**Mkdir:** crea un directorio nuevo.

EJ: mkdir directorioNuevo

Para crear directorios dentro de directorios q aun no están creados se usa el parametro **-p**

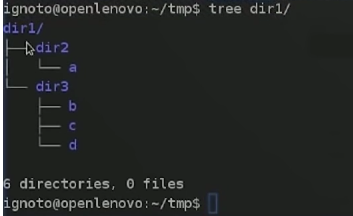
EJ: mkdir -p directorio2/directorio

El parámetro **-m** le da permisos al directorio.

Tambien podemos crear varios directorios bajo un mismo padre con un solo comando recursivo agrupando los directorios entre **{}**

EJ: mkdir -p dir1/{dir2/a,dir3/{b,c,d}}  
En este caso dir 1 tiene dir2 y dir3. Dir2 tiene a y dir3 tiene b, c y d

**Tree:** Este comando nos deja ver el árbol de un directorio.  
EJ: tree dir1



**Rmdir:** Solo borra directorios vacios

(usa parámetros similares a mkdir) EJ: -p

**Ls:** lista el contenido de ficheros. Tiene muchos parámetros.  
-l : muestra detalles sobre los archivos

-s: muestra tamaño en Kb

-d: muestra propiedades de un directorio pero no su contenido

-h: muestra el tamaño de directorios y ficheros en Gb Mb Kb según corresponda

-a: muestra todos los ficheros incluidos los ocultos (empiezan con “.”)

-R: Lista los recursos de forma recursiva (notar “R” en mayusculas)

EJ: ls -lh

**Permisos**

Un fichero es un archivo

los ficheros tienen 3 identidades

individual grupo y del resto del sistema

cada uno puede tener permisos de lectura escritura y ejecucion

el permiso de ejecucion para ficheros lo hace ser ejecutable

en cambio para los directorios o carpetas el permiso de ejecucion nos da acceso a entrar al directorio

Estos permisos e identidades se muestran de la siguiente manera.

d rwx r-x r-x

tenemos un bloque de 1 espacio y 3 bloques de 3.

En el primer espacio se indica el tipo de archivo siendo:

d: Directorio: contiene nombre de archivos que apuntan a inodos de disco.

l: Enlace simbolico. Este archivo contiene el nombre de otro archivo o directorio.

Es parecido a un ACCESO DIRECTO en windows

c: Dispositivo especial de caracteres. Un archivo que corresponde a un dispositivo de hardware el cual transfiere los datos en unidades de un bit.

p: Canal. Permite tener disponibles 2 protramas de linux y que se puedan comunicar uno con el otro. Uno abre el canal para leer y otro para escribir. permitiendo la trasferencia de datos.

s: Socket. es similar al llamado canal (p) pero este permite redireccionar enlaces.

-:archivo convencional debe ser texto, ejecutable, graficos, etc

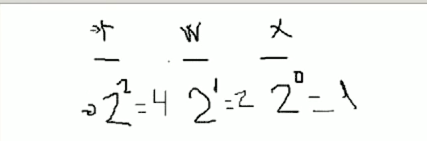
El primer grupo controla los permisos asociados al propietario del grupo, luego los del grupo del fichero o directorio, y el tercero al resto de los usuarios del sistema.

estos llevan r,w,x read write execute

lectura escritura y ejecucion.

un - muestra la ausencia de este permiso.

Para representar cada bloque en octalrealiza la siguiente cuenta.



Todos los valores habilitados nos darian 7 para un bloque.

**Super usuario**

Me hago super usuario con el servicio sudo escribiendo:

Sudo bash

Ahora puedo cambiar el propietario de un fichero con

Chown = change owner



Opciones pueden ser -R -C -f

R es de recursive cambia todos los ficheros dentro de un directorio.

C cambia el permiso para cada archivo

F no muestra errores si los hay

Chown root dir1

Chown -R root dir1

De la misma manera se puede cambiar permisos a un grupo con chgrp = change group

Con exit dejo de ser super usuario o root (salgo del servicio sudo).

No necesito ser super usuario para manejar los permisos de los archivos de los que soy propietario con

Chmod.

Se puede poner chmod 600 o chmod o-r

A other le eliminamos la escritura.

También se le puede agregar recursividad como a chown.

**Find**

Comando de búsqueda.

Find directorio filtro

Ej: find /home/Ubuntu/ -name dir

Find /home/usuario/ -perm 200

Find /home/usuario/ -perm \u=w

En el Segundo y tercer caso busco los que tengan permiso de solo escritura para usuario únicamente. Con la barra invertida y las letras u g o a y el filtro

Los filtros pueden ser.

-name

-perm

-size +40M 40 -5k

En el siguiente ejemplo devuelve un archivo a partir de un string en si mismo

find /<PATH> -type f -exec grep -H '<string>' {} \;

**Whereis**

Whereis

Busca ejecutables y devuelve primero donde se encuentra el ejecutable, luego su fichero de configuración y por ultimo donde esta ubicado su manual.

Es bueno para buscar la ruta absoluta para checkear errores de librerias por ejemplo.

Whereis ssh

Whereis python

**Cat**

Muestra el contenido de archivos

Ej: cat prueba.txt

Cat -E muestra el final de una línea mostrando $

Cat -T muestra las tabulaciones dentro de un texto

**Head**

Muestra las primeras 10 lineas de un archivo

Head syslog.1

Head -n 20 syslog.1

**Tail**

Muestra las 10 ultimas líneas

Tail syslog.1

Tail -f syslog.1 tail sigue funcionando y muestra en tiempo real lo que se va agregando al syslog.1

**Sort**

Ordena de forma alfabetica un fichero

Sort syslog.1

Sort -n syslog.1 ordena de forma numérica

Sort -r syslog.1 invierte el orden

**Xargs**

Se usa con el comando| para hacer que el resultado de la primer tubería se pase como argumento al final de la siguiente.

EJ: pidof htop | xargs kill

Esto es igual a kill 4902

Sin el xargs hubiera escrito 4902 kill

**Entradas y salidas de flujos de trabajo**

Existen 3 tipos

**Entrada estándar**: standard input (**stdin**)

Es por donde el usuario ingresa datos (teclado)

**Salida estándar**: standard output (**stdout**)

Es por donde el programa le da datos al usuario (pantalla)

**Error estándar**: standard error (**stderr**)

Es la salida de errores

Estas salidas pueden ser redireccionadas para que no aparezcan por pantalla y por ejemplo guardarlas en otro archivo de texto. Esto se hace con los siguientes comandos.

“>” redirige el contenido a otro archivo pisándolo

>> redirige el contenido a otro archivo añadiéndolo

2> redirige el error a otro archivo pisándolo

2>> redirige el error a otro archivo añadiéndolo

&> redirige tanto stdout como stderr

< redirige el contenido del fichero a la entrada (stdin)

<< añade el contenido del fichero a la entrada (stdin)

“<>” usa un fichero para entrada y salida

**Procesos y trabajos**

Los procesos se ven con el comando PS

Este muestra los distintos PID o identificadores de procesos.

EJ: ps fax

Ps faux este agrega el usuario, cpu y memoria utilizadas

El comando top te muestra los procesos de forma dinámica

Sudo atp-get install htop

Para ver los pid de un programa en particular uso

Pidof programa.

EJ: pidof Chrome

**Kill**

El commando kill se utiliza para matar procesos…para que estos terminen de guardar todo lo necesario se usa la señal (signal).

SIGTERM

EJ: kill -SIGTERM 4902

Para cerrarlo de forma brusca ponemos SIGKILL

Killall mata todos los procesos sin especificar el PID.

EJ: killall Chrome

**Tareas en primer y Segundo plano (jobs)**

Un job es un proceso parado.

Con Ctrl z detengo el proceso y lo deja como un job.

El comando Jobs muestra la cantidad de trabajos ejecutándose en el sistema

Forgrownd significa en primer plano. (fg)

Back ground significa en segundo plano.(bg)

Luego de parar el proceso lo puedo volver a ejecutar en primer o segundo plano indicándolo con el numero que figura al usar el comando Jobs

EJ: bg 1

Htop &

El & denota que el job esta ejecutándose en segundo plano.

**Gestion de usuarios**

**Root:** Super usuario admin. Unica cuenta q tiene acceso a todos lados tareas de mantenimiento instalar software.

UID=0 GID=0

User ID y Group ID

**Cuentas de sistema:** Tiene todos los privilegios para el servicio asociado. Normalmente no tienen contraseña.

UID y GID del 1 al 100

**Perfil de usuario:** Cada uno tiene un directorio de trabajo (/Home) y dentro de este tiene todos los privilegios

UID y GID del 500 en adelante dependiendo del software.

Estos datos se guardan en etc/Login.defs

Cat /etc/passwd tiene una estructura que nos da:

Usuario:password:UID:GID:comentario:Home del usuario:Shell que utiliza

Podemos cambiar la parte de bin/false por bin/bash para trabajar con algún usuario de sistema de forma temporal.

En etc/shadow están las passwords de los usuarios encriptadas.

**Super Usuario**

Con su cambiamos el usuario

Siempre se usa “su – “ para hacernos super usuario

Nos pide contraseña de root

El comando sudo nos permite ser super usuarios para realizar comandos concretos y no pide contraseña de root

Este listado de acciones se guarda en el fichero etc/sudoers. Este fichero se ve con el comando

Visudo.

Aquí podemos dar privilegios a un usuario en concreto para que maneje comandos concretos como super usuario y los otros no.

EJ: www-data ALL=ALL:NOPASSWD /etc/init/apache2

De esta forma cuando el usuario www-data utiliza sudo es super usuario, no le pide contraseña y puede utilizar el reinicio de apache2 como super usuario. Para el resto de las tareas no es super usuario

**Instalación de software**

**Apt:** es una herramienta para descarga e instalación de software.

La configuración de todos los programas instalados por apt están en **etc/apt/apt.config**

Los repositorios de donde descarga estos archivos se encuentra en:

**/etc/apt/source.list**

para buscar un programa usamos el comando

apt-cache search

EJ: apt-cache search libreoffice

Con el commando apt-get tengo varias opciones.

Apt-get update: actualiza la base de datos de los repos. (source.list)

Apt-get upgrade: actualiza los paquetes q es seguro q no traerán conflictos. Es recomendable reiniciar después de esto por si hay una actualización de kernel.

Apt-get dist-upgrade:realiza una actualización completa resolviendo conflictos. Es recomendable hacer esto después de un apt-get upgrade

Apt-get install: instala el programa nuevo

EJ: apt-get install iptraf

Apt-get remove: elimina el programa

EJ: apt-get remove iptraf

Apt-get remove --purge: elimina Tambien los ficheros de configuración de este programa

EJ: apt-get remove --purge iptraf

Apt-get clean: borra los paquetes descargads en cache de instalación (/var/cache/apt)

Apt-get autoclean: borra los paquetes inservibles en cache

Apt-get autoremove: elimina los paquetes ya instalados como dependencia y q su paquete padre fue eliminado.

**Cron**

Permite realizar tareas programadas.  
el usuario genera tareas programadas con **crontab**

**Crontab:** sigue la siguiente sintaxis.  
**m** h **dom** mon **dow** user command

**Min** hora **dia de forma numérica del mes** mes **días de la semana** (siendo 0 y 7 dom o las primeras 3 letras en ingles)

Con **crontab -l** (list)me muestra una lista de las tareas programadas en crontab para mi usuario en curso.

Con **crontab -e** (edit)me deja crear una tarea

Con **crontab -r** (remove) elimina la tarea

Fuente: https://www.youtube.com/watch?v=mvuVvsR5J9A&list=PLf8XMtbjh0dVXtOPb46WXbThG5xB-Xffu&index=27