Apuntes LPIC1

Tema 101: Arquitectura de sistema:

1) Identificar y editar configuraciones de hardware:

Activación de dispositivos:

El hardware básico se configura desde la BIOS.

Inspección de dispositivos:

Comandos:

- **lspci**: muestra todo lo conectado al bus pci:
  - -s id: muestra la info de ese dispositivo.
  - -v: muestra todos los detalles.
- **Lsusb**: muestra los dispositivos usb conectados:
  - -d id: muestra la info de ese dispositivo.
  - -v: muestra detalles.
- **Lsmod**: muestra los módulos (drivers) cargados actualmente.

Los **datos** del **hardware** están **almacenados** en **/proc y /sys**, mediante cat podemos mostrarlo.

Tipos de dispositivos:

- **Coldplug**: es necesario apagar el pc para conectar el dispositivo, como tarjetas PCI, dispositivos IDE, etc.
- **Hotplug**: permite conectar/desconectar en caliente, ej: usb. Cuando se conecta un dispositivo hotplug automáticamente se activa un evento Udev que actualiza los dispositivos en **/dev**.

Dispositivos de almacenamiento:
Cualquier dispositivo de almacenamiento se identifica en un archivo dentro de /dev. El nombre del archivo depende del tipo de dispositivo (SATA, IDE...) y de las particiones.

/dev/fd0: disquetera

Tipo	Criterio de nombramiento	Ejemplo
IDE	Canal IDE usado, Master/Slave y particiones	**/dev/hda1**: 1ª partición del primer canal master. /dev/hdb2: 2ª partición del primer canal slave /dev/hdc3: 3ª partición del segundo canal master.
SATA	Por la BIOS y particiones	**/dev/sda2**: 2ª partición del primer disco. /dev/sdb1: 1ª partición del segundo disco
SSD (Pendrives, etc)	Por la BIOS y particiones	**/dev/sdc1** (como si fuese sata /sda...)

2) Inicio (boot) del sistema:

Es posible pasarle parámetros al kernel en el momento de inicio para especificar datos como memoria, dispositivos, etc.

Cargador de boot (Bootloader):
Los principales son Grub y Lilo, a estos también se les puede añadir parámetros como:

Parámetro	Descripción	Ejemplo
acpi	Conecta/desconecta el soporte ACPI	acpi=off
init	Define otro programa en lugar de /sbin/init	Init=/bin/bash
mem	Define cuanta ram estará disponible	mem=512
maxcpus	Nº máximo de núcleos.	maxcpus=2
quiet	No muestra la mayoría de los mensajes de inicio	quite
vga	Modo de video	vga=773
root	Define partición raiz diferente a la del bootloader	root=/dev/sda3
ro o rw	Realiza el montaje en sólo lectura	ro

También podemos modificar el runlevel inicial, todo se puede configurar desde grub y lilo.

Mensajes de inicio:
Para mostrar los mensajes de inicio quitaríamos **quite y splash** de la línea de carga del kernel. **Para inspeccionar** el proceso de inicio usamos **dmesg** y observaríamos los mensajes de **/var/log/dmesg**.

3) Cambiar runlevels, apagar y reiniciar el sistema

El runlevel es el nivel de ejecución del sistema, el programa **/sbin/init** que es invocado **tras** el **bootloader identifica** e **nivel de ejecución** metido **como parámetro en la carga del kernel o mirandolo** en **/etc/inittab**. Luego carga los programas, script y servicios indicados en ese archivo. La mayor parte de los **script ejecutados por init** se **guardan** en **/etc/init.d** pero en otras distros es en **/etc/rc.d**.

Los niveles de ejecución:
Se enumeran del 0 al 6 y tienen el formato:

- **id:** nombre de hasta 4 caracteres para identificar la entrada en inittab.
- **runlevels**: es la lista de runlevels para los que se ejecutará.
- **acción**: el tipo de acción: sysinit (inicio del sistema), wait (se ejecuta y se espera su finalización) y ctrlaltdel (se ejecuta cuando se pulsa control+alt+del o se reciba señal SIGINT).
- **proceso**: comando que se activará.

Numeración de los runlevels:

- 0: Apagar el sistema.
- 1: usuario único, sin red o servicios
- 2: multiusuario, estándar en la mayoría de distros.
- 3: multiusuario, estándar en otras distros.
- 4: no se usa.
- 5: no se usa
- 6: reinicio del sistema.

Para **cambiar** entre **runlevels** se usa **telinit nº_runlevel** y para **ver** en **que runlevel** estamos ejecutando el so escribimos **runlevel**.

Apagar y reiniciar:
A, apagar con el comando shutdown se notifica a todos los usuarios del so que se va a apagar, al apagar se pasaría al runlevel 1 de usuario único. El comando lo usamos como:
shutdown opción horario mensaje

El horario puede ser:

- hh:mm hora de ejecución.
- +m: minutos para la ejecución.
- Now o +0: ejecución inmediata

Opciones:
-a: usar el archivo de permiso /etc/shutdown.allow.
-r: reiniciar la máquina.
-h: apagar el pc.
-t segundos: tiempo de espera antes de que se ejecute la acción.

El mensaje será enviado a todos los usuarios.

Para prevenir que cualquiera pueda reiniciar la máquina con ctrl+alt+del añadimos la opción -a en el comando shutdown de la línea ctrlaltdel de /etc/inittab.

Tema 102: Instalación de Linux y administración de paquetes:

1) Dimensionar particiones de disco:

En Linux se accede a todos los sistemas de archivos en particiones por medio del montaje, en el cuál ese sistema queda "montado" en un directorio (punto de montaje).

Sistema de archivos raíz:
El **principal punto de monta es raíz "/**" que es el primer punto de montaje que tendrá el equipo, esta partición queda **identificada con el cód hex 83 (0x83 Linux Native)** y ya podrán copiarse los archivos del sistema.

Orden d montaje a partir del boot:
El **cargador de boot (Grub o Lilo) carga el kernel y transmite** información sobre la **localización** de **/**. **Se monta** la raiz y **los demás** dispositivos **se montan de acuerdo a** las instrucciones de **/etc/fstab** (debe de estar en / porque si no no podrá montar nada).

Linux exige como mínimo dos particiones: raíz y swap.

La partición Swap:
Es un espacio en disco que actua como memoria adicional evitando la ocupación total de la RAM. Como norma general se suele crear una partición swap del doble de tamaño que la RAM, pero ya no se usa en sistemas con varios gb de ram. Es recomendable crearla en los dispositivos más veloces.

Otros puntos de montaje:
Aunque todo puede alojarse en / **podemos crear particiones diferentes para algunos directorios**, esto ocurre en servidores con mucha demanda.
En otras particiones pueden estar:

/var	Contiene cola de mails, impresión y DB que son bastante usadas. También los logs.
/tmp	Espacio temporal usado por programas, si se pone en otra partición evitamos colapso del directorio raíz /.
/home	Contiene los datos personales de usuarios, en otra partición limitamos el espacio.
/boot	Contiene el kernel y archivos de bootloader Grub, es necesario cambiarlo a otra partición si la máquina requiere que el kernel esté antes del cilindro 1024 del hdd y cuando el bootloader no es capaz de trabajar con el sistema de archivos de /.
/usr	Programas, código fuente y documentación. En otro dispositivo reduce la intensidad de acceso al mismo.

Algunos directorios no deben estar fuera de /: /etc, /bin, /sbin, /dev, /proc y /sys.

2) Instalar el gestor de arranque:

El bootloader es el responsable de **localizar y cargar el kernel** y desempeaña una **etapa intermedia entre** el final de procedimientos de la **BIOS y el inicio del SO**.
Después de terminar los chequeos de la **BIOS, carga en memoria los datos del MBR** del disco definido como inicio en la BIOS. **El MBR** (Master Boot Record) ocupa el primer sector del disco (512 bytes), que **contiene la tabla de particiones y el cargador de arranque**. En una de estas dos últimas puede estar **el kernel, lo localiza y lo carga**. El cargador más usado es Grub y Lilo.

Lilo:
El Lilo o Linux Loader es el cargador más tracicional y actualmente sólo lo usa Slackware. Se divide en:

- **Lilo**: el cargador, se instala en la MBR y activa la segunda parte del cargador de arranque localizado en /boot/boot.b.
- **/etc/lilo.conf**: archivo de configuración.
- **/sbin/lilo**: comando que lee las configuraciones e instala el cargador en la MBR.

Lilo guarda en la MBR el cargador de boot, la información de localización del Kernel y el directorio raiz.
Opciones de /etc/lilo.conf:

boot	Donde está el cargador de lilo, en el MBR de /dev/hda o /dev/sda.
prompt	Muestra al usuario un menú para seleccionar el SO.
image	Localización del archivo del kernel.
other	Indica la partición de otro SO.
label	Nombre para mostrar en el menú de inicio.
root	Indica la partición raíz para el kernel indicado.
read-only	Partición montada solamente como lectura en la primera etapa del boot.
append	Parámetros adicionales del kernel.
message	Indica mediante un archivo de texto el mensaje que se mostrará en el menú de boot.
delay	La pusa en décimas de segundo para que el user pulse Tab y active el menú.
timeout	Define la pausa en décimas de segundo para que se cargue la opción estandar o seleccionada.
vga	Valor numérico que especifica las preferencias visuales del terminal.

Después de modificar el archivo /etc/lilo.conf es importante **resintalar** el cargador del MBR **con el comando lilo**.

Grub:
El Grub (Grand Unified Bootloader) **es el más usado**, también está **instalado en la MBR (/sbin/grub-install)** y carga las **configuraciones de /etc/grub/menu.lst**.

Opciones generales de /boot/grub/menu.lst:

- **default**: SO que iniciará, por defecto 0.
- **timeout**: tiempo de espera en seg para iniciar el boot.

Opciones individuales:

- **title**: Nombre para el ítem.
- **root**: localización del cargador de la segunda etapa y del kernel (/dev/sda o /dev/hda).
- **kernel**: camino para el kernel.
- **ro**: montar inicialmente en sólo lectura.
- **initrd**: camino para la imagen del initrd.

La ventaja de Grub sobre Lilo es que no hay que reinstalar en el MBR cada vez que se modifica algo. Nota: en Grub2 cambia y hay que actualizar con sudo update-grub2.

Dispositivos de inicio alternativo:
Es posible iniciar el sistema con un liveCD y modificar esos archivos.
Una buena práctica es hacer **backup del MBR** con:
dd if=/dev/hda of=mbr.backup bs=1 count=512
Podemos restaurar esta copia con:
dd if=mbr.backup of=/dev/hda

3) Control de bibliotecas compartidas:

Funciones comunes de diferentes programas **se almacenan en bibliotecas**. **Para compilar** un programa **es necesario** que pueda **localizar las bibliotecas que necesita** y así crear vínculos entre sus funciones y las de las bibliotecas.
Este vínculo p**uede ser estática** (quedan **incluidas en el programa** compilado) **o dinámico** (quedan **mapeadas** a la biblioteca externa).

Identificar bibliotecas compartidas:
Para **conocer las bibliotecas necesarias por un programa** se usa el comando **ldd**, ejemplo: ldd /usr/bin/vi y mostrará todas las bibliotecas que necesita. Si aparece not found es que hay un error y no encuentra esa biblioteca.

Localización de las bibliotecas
El **programa** responsable de **cargar la biblioteca y vincularla** al programa **es ld.so** que es invocado cada vez que un programa necesita una función de biblioteca externa.
El **ld.so consigue localizar con** la ayuda del mapeo encontrado en **/etc/ld.so.cache**.
Las **blibliotecas** estandar **del sistema suelen encontrarse en /lib y /usr/lib**, **si se añaden otros** directorios **se deben de incluir en /etc/ld.so.conf o en /etc/ld.so.conf.d/** en algunas distros. El **comando ldconfig actualiza esta caché** con las configuraciones de ld.

4)Utilización del sistema de paquetes Debian:

Permite la instalación sin que el usuario se preocupe por las bibliotecas u otros programas necesarios para la ejecución del mismo ya que cada .deb contiene información de sus dependencias.

Las principales herramientas de administración de paquetes .deb son:

- **dpkg**: para instalar paquetes individuales.
- **apt-get**: busca paquetes en repositorios remotos y los instala además de sus dependencias.
- **aptitude**: alternativa a apt-get.

La gran **ventaja** es que las **dependencias** de ese programa a instalar, **las baja automáticamente** y las instala sin problemas.

Repositorios:
Para la resolución automática de dependencias es **necesario indicar el origen de los paquetes**, estos orígenes estan determinados **en /etc/apt/sources.list y** en algunos casos en **/etc/apt/sources.list.d/.**
Cada línea determina un repositorio y cada distribución tiene repositorios propios oficiales y no oficiales. Después de actualizar los orígenes, es **necesario actualizar con apt-get update o aptitude update** para que se actualice la caché.

Instalación:
Para **buscar programas** podemos usar **apt-cache search nombre_paquete o aptitude search nombre_paquete**, no es necesario el nombre exacto.
Para **instalar** usamos **apt-get install nombre_paquete o aptitude install nombre_paquete**.
Si lo que queremos es **instalar un .deb** usamos **dpkg -i nombre_paquete.deb**, si es necesario **reconfigurar** el paquete **después de la instalación** ejecutamos **dpkg-reconfigure nombre_paquete**.

Eliminación:
Para **desinstalar** programas usamos **apt-get remove nombre_paquete o aptitude remove nombre_paquete**. Para **eliminar** también los archivos de **configuración de ese programa usamos: apt-get remove --purge nombre_paquete**.

Actualizar programas:
Los paquetes se actualizan con **apt-get upgrade.**

Inspección de paquetes:
Con **dpkg**:

- **-l nombre_paquete**: muestra el estado del paquete.
- **-S nombre_archivo**: busca que paquete instaló ese archivo.
- **-L nombre_paquete**: lista los archivos instalados por un paquete.
- **--contents paquete.deb**: lista el contenido del paquete seleccionado.

También usamos **apt-cache show nombre_programa**: **muestra la descripción y detalles del paquete** seleccionado.

5) Utilización del sistema de paquetes RPM y YUM

Es semejante al sistema de paquetes de Debian.

RPM
Es el comando de administración de paquetes rpm, **semejante a dkpg**.
Podemos **instalar un paquete .rpm con rpm -ivh nombre_paquete.rpm**. Como **opciones** tenemos:

- **-i o --install**: instala paquete.
- **-U o --update**: actualiza paquete.
- **-F o --freshen:** actualiza el paquete si estuviese instalado.
- **-V o --verify**: verifica tamaño, MDB, permisos, etc.
- **-q o --query**: investiga paquetes y archivos.
- **-e o --erase**: desinstala el paquete.
- **--nodeps**: instala sin verificar dependencias
- **--force:** fuerza instalación/actualización.
- **--test:** muestra como sería la instalación pero no la hace.
- **--requires:** con opción ”q” muestra las exigencias del paquete.
- **--whatrequires:** con opción “q” muestra que programas dependen del paquete.

Hay subopciones que modifican su comportamiento, sobre todo con -q, ejemplo -qc: lista archivos de configuración.

Conversión y extracción:
Con el comando rmp2cpio mostramos el conteido del archivo rpm usamos rpm2cpio nombre_paquete.rpm | cpio -t.
Para extraer archivos rpm2cpio nombre_paquete.rpm | cpio -ivd '*pdf' extraermos todos los pdf del archivo.

YUM

Es **semejante** al comando **apt-get.** Su archivo de configuración **/etc/yum.conf**,, algunas opciones:

- **cachedir:** directorio de almacenamiento de los paquetes, por defecto **/var/cache/yum.**
- **keepcache: 1 mantiene** los archivos tras la instalación o **0 no los mantiene**.
- **reposdir:** lista de **directorios** donde yum buscará **repositorios**, por defecto **/etc/yum.repos.d**.
- **debuglevel**: nivel del mensaje de aviso del 0 al 10, por defecto 2.
- **errorlevel**: nivel del mensaje de error del 0 al 10, por defecto 2.
- **logfile**: ruta al log de yum.
- **gpgcheck**: determina si yum debe verificar GPG, 1 o 0.

Funciones del comando yum:

- **search nombre_paquete**: localiza un paquete por el termino buscado.
- **install nombre_paquete**: instala el paquete.
- **remove o erase nombre_paquete**: desinstala paquete
- **provides o whatprovides recurso**: localiza que paquete ofrece un determinado recurso.
- **update: actualiza los paquetes**.
- **upgrade: igual que updat**e pero también actualiza la distribución.

Con **yumdownloader** nombre_paquete podemos **descargarlo sin** tener que **instalarlo.**

Tema 103: Comandos GNU y Unix

1) Trabajar en la línea de comandos

Es un prompt del shell de usuario para introducir instrucciones. Si termina con $ indica que es un usuario común y si es # es el usuario root.

El shell Bash

Es el estándar en la mayoría de las distros Linux, significa Bourne Again Shell.

Algunos comandos importantes incorporados del bash.

Comando	Finalidad	Ejemplo
alias	Crea un alias para otro comando	alias rm='rm -i'
exec	Un comando invocado con exec sustituye a la sesión actual de shell	exec telinit 1
echo	Imprime un texto o variable	echo $PATH
env	Sin argumentos muestras las variables de entorno y contenidos	env DISPLAY=:1.0 xclock
export	Define una variable de entorno para la sesión y todas a partir de esta	Export PATH=$PATH:/usr/...
pwd	Muestra el directorio actual	pwd o echo $PWD
set	Define una variable, sin argumentos muestra las definidas	Set NOMBRE='Monino'
unset	Elimina una variable en la sesión	unset Nombre

Variables

No deben de tener espacios y para crearlas es: nombre_variable=valor, para mostrar el valor de la variable usamos $, ejemplo: echo $nombre_variable.

Hay dos tipos:

- **Locales**: sólo accesibles en la sesión actual de shell.
- **Exportadas**: accesibles en la actual y en las siguientes.

Variables predefinidas:

Variable	Definición
DISPLAY	Determina en que display de X el programa debe mostrar sus ventanas.
HISTFILE	Ruta al historial de comandos de usuario ($HOME/.bash_history).
HOME	Ruta al directorio personal del usuario.
LOGNAME	Nombre que el usuario usó para entrar al sistema.
PATH	Lista de directorios en los cuales se buscarán programas.
PWD	El directorio actual.
SHELL	El shell usado.
TERM	Tipo de emulador, es distinto si se usa en X que en consola tty.

Otras variables incorporadas actúan casi como comandos, devolviendo valores:

- **$!**: PID del último proceso ejecutado.
- **$$**: PID del shell actual.
- **$?**: Devuelve 0 si el último comando tiene éxito, si no devuelve 1.
- **~**: Directorio personal del usuario actual.
- **~alaor**: Directorio personal del usuario alaor.

Comandos secuenciales

Para ejecutar comandos unos tras otros podemos usamos este formato:

comando1 ; comando2 ; comando3

Si queremos ejecutar arios comandos solamente si el anterior fue correcto usamos:

comando1 && comando2 && comando3

Para ejecutar el comando sólo si el anterior no tuvo éxito usamos:

comando1 || comando2 || comando3

Referencia y manuales

Tras escribir las primeras letras de un comando o directorio, con la tecla **tabulador** podemos **completar la palabra**.

Para **consultar** el **manual** de cada comando usamos: **comando man**, **también** podemos usar **info** pero da menos información. Estos manuales tienen la siguiente organización:

- **Nombre**: asunto de la página del manual y breve descripción.
- **Sinopsis**: la sintaxis del comando.
- **Descripción**: descripción detallada.
- **Opciones**: revisión de todas las opciones y sus funciones.
- **Archivos**: archivos relacionados con el asunto.
- **Vea también**: otras páginas relacionadas.

Es posible **buscar ocurrencias** de un termino **en** la sección nombre de **los manuales** con el **comando apropos**, devuelve una breve descripción y el nombre del comando.

Podemos **identificar todos los manuales que hacen referencia a un** determinado **término** con **whatis**.

Los archivos de manualse se **almacenan** en **/usr/man** y **/usr/share/man**, pero podemos añadir más rutas configurando **/usr/lib/man.conf** o **/etc/man.conf**.

Impresión de manuales

Se pueden imprimir como texto sin formato direccionando la salida del comando man. **Pero** p**odemos dejarle el formato con** el comando **groff**, ejemplo:

zcat /usr/man/man1/find.1.gz | groff -man -Tps > find.ps

o imprimir directamente con:

zcat /usr/man/man1/find.1.gz | groff -man -Tps | lpr

2) Procesar cadenas de texto por medio de filtros

- **cat**: Se usa para mostrar el contenido de archivos y pude actuar como redireccionador.
- **tac**: igual que cat pero muestra el contenido de atrás hacia delante.
- **head**: muestra el inicio de los archivos, por defecto 10 primeras líneas, pero con la opción -n podemos indicar cuantas líneas y con -c indicamo el número de caracteres que se mostraran.
- **tail**: igual que head pero muestra el final, tiene las mismas opciones pero además tiene -f que muestra continuamente el final cuando se agrega más texto.
- **wc**: cuenta líneas (-l), palabras (-w) o caracteres (-c), sin argumentos muestra las 3.
- **nl**: número de líneas como el comando cat -b. Con el argumento -ba se numeran todas las líneas. Con -bt se numeran las líneas que no están en blanco.
- **expand**: sustituye espacios tabulados por espacios simples manteniendo la distancia.
- **unexpand**: sustituye dos o más espacios simples por una tabulación.
- **hexdump**: nuestra archivos binarios, con -c es más legible.
- **od**: se usa para convertir entre diferentes formatos, hexadecimal, binario, etc.
- **split**: divide un archivo en otros menores, con -l se indica el número de líneas de cada parte, con -b el tamaño de cada parte . Ejemplo: split -b 1024k archivo_original parte_ y creará parte_aa, parte_ab.... Para volver a concatenarlo usamos cat parte_* > archivo_entero.
- **uniq**: muestra el contenido de archivos quitando lineas repetidas, con -u muestra sólo las líneas que no se repiten.
- **cut**: delimita un archivo en columnas, para serparar por cambo usamos -d carácter_delimitador y -f informa la posición del campo. Ejemplo mostramos los campos 1 y 3 del archivo /etc/group delimitados por :, usamos: cut -d ':' -f 1,3 /etc/group. Para indicar otro delimitador en lugar del original usamos –output-delimiter 'carácter'.
- **paste**: concatena archivos lado a lado en forma de columnas. Ej: paste texto1 texto2 y mostraria linea1del1 linea1del2, etc.
- **join**: paste pero especificando campos, join -1 CAMPO -2 CAMPO archivo1 archivo2.
- **sort**: ordena alfabéticamente, con -n lo hace numéricamente. Con -r invierte el resultado.
- **fmt**: configura un texto para determinado número de caracteres por linea, el estándar es 75. Con -w indicamos el nº de caracteres por línea, con -s divide líneas grandes y -u un espacio entra palabras y dos entra sentencias.
- **pr**: divide el archivo para impresión, el estándar es 66 líneas (-l) de 72 caracteres (-w).
- **tr**: convierte caracteres, ej: echo abc | tr '[a-z]' '[A-Z]', podemos sustituir espacios por otro carácter con: echo 'Frase con espacios' | tr ' ' '_'.

3) Administración básica de archivos

Directorios y archivos

Pueden accederse tanto por su camino absoluto (/../....) como por el relativo. El punto . es e directorio actual y los dos puntos .. es el directorio superior al actual.

El comando ls lo usamos para listar archivos y contenido e un directorio, con -l se muestran más detalles, -s muestra el tamaño en KB.

El comando **ls -l** muestra de izquierda a derecha:

- Tipo de archivo y permisos.
- Número de hardlinks para el archivo.
- Dueño
- Grupo al que pertenece.
- Tamaño en bytes.
- Fecha de la última modificación.
- Hora de la última modificación.
- Nombre del archivo.

Con el comando **file** podemos saber que **tipo de archivo** es.

Manipulación de archivos y directorios

El comando **cp** se usa para copiar archivos y como argumentos tiene:

- **-i**: modo interactivo, va preguntando.
- **-p**: copia tambien los atributos del archivo original.
- **-r**: copia recursivamente el contenido del directorio de origen. Si ponemos la barra / al final del directorio de origen se copiará sólo el contenido al directorio de destino, si no la ponemos se copiará el contenido y el directorio.

Con el comando **mv movemos y renombramos archivos**, con -i pide confirmación.

Con **touch** podemos **modificar** la **fecha** de **creación** de un archivo a los valores actuales del sistema, Para sólo modificar la fecha usamos -m y para la hora -a. Otros valores con -t.

Navegamos entre directorios **con** el comando **cd**.

Para **crear carpetas** usamos **mkdir**, podemos crear una **ruta entera con** todos sus directorios con **-p**, ejemplo: mkdir -p /home/monino/noexisto1/noexisto2. Con la opción -m modificar los permisos a la hora de crearlo usamos -m.

Para **borrar directorios** usamos **rmdir**, pero para **borrar archivos** usamos **rm**. Si añadimos la opción -p se borrará un árbol de directorios si están vacios, para borrar con contenido sería -r y para forzar la eliminación usamos -f.

Empaquetar archivos

Existen d**os comandos: tar y cpio** y **sirven** para **agrupar en un archivo**. Para crear un archivo que contiene un directorio usamos: **tar cvf nombre_agrupado /directorio**, con c crea un archivo, con v muestra cada archivo incluido y con f especifica el camino al archivo creado. **Para extraer usamos xvf en vez de cvf.**

Los **comandos de compresión son gzip y bzip2**, para comprimir usamos **gzip** archivo **o bzip2 archivo**, esto creará automáticamente un archivo **.gz** o **.bz2**. La diferencia es que **gzip es más rápido pero bzip2 comprime más.**

Podemos **comprimir directamente al pasar a tar con: tar czvf nombre_agrupado.gz /directorio**, de esta manera **usaremos gzip**, **para** usar **bzip2** escribiríamos **cjvf** e**n vez de czvf**.

Para **descomprimir** usamos **gunzip y bunzip2** y también puede redireccionarse con el comando **tar con** las opciones **z y j.**

La diferencia de tar con cpio, es que **cpio trabaja con redireccionamiento de entrada y salida estándar.**

Caracteres comodín (file globbing)

- *****: sustituye a cualquier secuencia de caracteres, mientras que ? Sólo sustituye a uno.
- **[]**: indica una lista de caracteres, ejemplo: [abc].
- **{}**: indica una lista de términos separados por coma, ejemplo: ls /dev/{hda,fd0}.
- **!**: Antes de un carácter comodín lo elimina de la operación.
- \: antes de cualquier comodín no realizan sustitución.

Búsqueda de archivos

El comando principal es **find directorio criterio**, el directorio es donde debe iniciar la búsqueda y el criterio el nombre del archivo o directorio o una regla para búsqueda, criterios:

Criterio	Definición
-type	Define el tipo: archivo (f), directorio (d) o enlace (l).
-name nombre	Nombre del archivo.
-user usuario	Dueño del archivo.
-atime -/+n	Accedido antes o después de n (n*24 horas).
-ctime -/+n	Archivo creado antes o después de n.
-mtime -/+n	Archivo modificado antes o después de n.
-amin -/+n	Archivo accedido antes o después de n. (n es la cantidad en minutos)
-cmin -/+n	Archivo creado antes o después de n.
-mmin -/+n	Archivo modificado antes o después de n.
-newer archivo	El archivo buscado se creó o modificó después de archivo.
-perm modo	El archivo buscado tiene permiso igual a modo (r,w,l).
-perm -modo	El archivo buscado tiene todos los permisos de modo.
-perm +modo	El archivo buscado tiene cualquiera de los permisos de modo.

Ejemplo para encontrar todos los enlaces en /usr/lib creados hace menos de 24 horas:

find /usr/lib -type l -ctime -1

4) Flujos pipes (tuberías) y redireccionamiento de salida

Por defecto el descriptor de entrada (stdin) es el teclado, el de salida (stdout) y error (stderr) son la pantalla.

Redireccionamiento

Para **redireccionar** la salida estándar de un comando a un archivo usamos **>**, ejemplo:

cat /etc/passwd > copia_pass, **si el archivo existe se sobreescribirá**, **para agregar valores sin borrar el anterior usamos >>.**

Para redireccionar el contenido de un archivo a la entrada estándar de un comando usamos <.

Para redireccionar stderr usamos 2> y para stderr+stdout usamos &>.

Tubería (pipe)

Con | enviamos la salida de un comando a la entrada de otro.

También podemos redireccionar la salida simultáneamente tanto para un archivo como par stdout con el comando tee. Ej: cat /etc/passwd | tee copia_pass Mostrará por pantalla y almacerá.

Sustitución de comandos

Es posible usar la salida de un comando como argumento para otro usando comillas invertidas ``. Ej: ls -l `cat /etc/ld.so.conf`.

Con el comando xargs pasamos los datos que recibe por stdin como argumento para otro comando, este funciona de intermediario.

5) Crear, monitorear y finalizar procesos

Un proceso es un programa en ejecución, cada proceso tiene un PID (número único de identificación).

Monitorear procesos

Pueden usarse varios procesos:

- **ps**: Muestra los procesos activos de manera detallada.
- **top**: monitorea continuamente los procesos, uso de memoria y CPU
- **pstree**: muestra procesos activos en árbol genealógico.
- **kill PID**: envia la señal SIGTERM con valor 15, que pide la finalización del programa.
- **killall**: funciona igual que kill pero usa el nombre del proceso en lugar del PID. Con -l se listan las señales posibles.

Tareas en primer y segundo plano

Normalmente se ejecutan en primer plano, para interrumpir el programa y volver al shell usamos Ctrl+Z. Para ejecutar el programa en segundo plano después de haberlo parado usamos el comando **bg** y para volver a primer plano **fg**.

Podemos iniciar programas directamente en segundo plano usando & al final del comando.

Recursos del sistema

Los procesos que ocupan mucha memoria o procesamiento deben finalizarse en situaciones de emergencia.

Con el comando **free** podemos ver: **cantidad total de ram, memoria libre y espacio de swap**.

Para identificar el consumo de recursos usamos **uptime** que **muestra el promedio de consumo** general, si aparece 1.00 está requiriendo mucho.

6) Modificar la prioridad de ejecución de un proceso

Como en todos los SOs multitarea se pueden atribuir prioridades a los procesos, estas **prioridades** se definen **con números** denominados **NI (números nice)**, usados para modificar la prioridad de CPU y balancear su uso.

Por defecto se inicia con prioridad **0**, los NI van **desde** la más **baja (19) a** la más **alta (-20)**, sólo root puede cambiarlo por debajo de 0.

Para **iniciar** un **comando** con prioridad cambiada **usamos nice**, ej: **nice -n nº_prioridad comando**, también es posible **modificar** la **prioridad** de un proceso en marcha con **renice**, ej: **renice -n nº_prioridad -p PID**.

Podemos **modificar todos los procesos del grupo o usuario** con:

renice +/-nº_prioridad -g/u nombre_grupo/usuario.

7) Buscar en archivos de texto usando expresiones regulares

Grep

Muchos programas soportan el uso de expresiones regulares.

Carácter	Finalidad
^	Inicio de línea.
$	Final de línea.
.	Cualquier carácter.
*****	Cualquier secuencia de 0 o más caracteres.
[]	Cualquier carácter que esté presente en los corchetes.

Un uso común de grep es mostrar el contenido de los archivos de configuración borrando las líneas que corresponden con los comentarios (#), ej: **grep '^#' /etc/lilo.conf.**

Ejemplo para mostrar las líneas que contenta hda o hdb: **grep 'hd[ab]' /etc/lilo.conf**.

Opciones de grep:

- **-c**: cuenta las líneas.
- **-i**: ignora la diferencia entre mayúsculas y minúsculas.
- **-f**: usa la expresión regular incluida en el archivo indicado por esta opción.
- **-n**: busca solamente en la línea indicada.
- **-v**: muestra todas las líneas, excepto la que corresponde al estándar.

Variaciones del grep

Son:

- **egrep**: es equivalente a grep -E, egrep puede usar operadores como pipe | que actua como O, ej: egrep 'invención|invenciones'.
- **fgrep**: actua como grep -F, con lo que deja de interpretar expresiones regulares. Ej: fgrep 'Sevilla'.

Edición de estándares con sed

Sed se usa **para buscar y sustituir estándares** en texto y mostrando el resultado en pantalla, su sintaxis es: **sed opciones 'comando y expresión regular' archivo**. **No modifica el archivo de origen.**

Ej: sed -e '/^#/d' /etc/lilo.conf, esto muestra el contenido de lilo.conf sin las líneas iniciadas con #.

Opciones de sed:

- **-e**: ejecuta la siguiente expresión y comando.
- **-f**: lee expresiones y comandos del archivo.
- **-n**: no muestra las líneas que no correspondan con la expresión.

Comandos de sed:

- **s**: sustituir.
- **d**: borra la línea.
- **r**: inserta el contenido del archivo indicado en la ocurrencia de la expresión.
- **w**: escribe la salida en el archivo indicado.
- **g**: sustituye toda las ocurrencias de la expresión en la línea actual.

8)Edición básica de archivos con VI

Está en la mayoría de distribuciones.

Modos de ejecución

Tiene tres modos de ejecución: navegación, inserción y comando.

Navegación

Es el modo inicia, las teclas del teclado actual para la navegación.

Modo de inserción

Pulsamos la tecla i o A para entrar en este modo, se usa para digitar texo. Con la tecla ESC volvemos al modo navegación.

Modo comando

Accedemos al pulsar : desde navegación, se usa para realizar búsquedas, guardar, salir, ejecutar, etc. Para volver a navegación pulsamos Enter en una línea vacía.

Tema 104: Dispositivos, sistemas de archivos y FHS

1) Crear particiones y sistemas de archivos

Para usar el disco e instalar Linux hay que particionar el hdd.

Fdisk

Fdisk es un programa para manipular particiones, con la opción -l mostramos las particiones existentes y los dispositivos. Para manipular una partición ejecutaremos fdisk dispositivo. Las particiones tienen un nº hexadecimal que indica el sistema de archivos: Linux 83 (0x83) y swap 82 (0x82).

Comandos fdisk:

- **-p**: lista las particiones.
- **-n**: crea una particion nueva interactivamente.
- **-t**: cambia el código de identificación de la partición.
- **-d**: borra una partición.
- **-q**: sale de fdisk sin guardar la partición.
- **-w** sale del fdisk y guarda las modificaciones.
- **-m**: muestra la ayuda.

Para **convertir ext2 en ext3** sin perder datos ejecutamos: **tune2fs -j /dev/dispositivo**.

Creación de sistemas de archivos

Con el comando **mkfs** podemos **formatear** las particiones, con la opción -t elegimos el sistema de archivos, ejemplo: **mkfs -t ext3 /dev/hda3**. Existen comandos específicos para cada sistema de archivos: mkfs (ext2, ext3, ext4, xfs y vfat), mke2fs, mkreiserfs y mkdosfs.

Partición swap

Podemos **formatear** la partición con **mkswap**, ejemplo: **mkswap /dev/hda2**. Luego tenemos que activarla para que se use con swapon -a, que activa toads las particiones swap encontradas en /etc/fstab. También podemos **desactivarla** con **swapoff dispositivo**.

2) Mantenimiento de la integridad de sistemas de archivos

Es recomendable efectuar un chequeo y corrección de las particiones

Chequeo del sistema de archivos

Para ello usamos **fsck** en las **particiones** que deben estar **desmontada** o montada en sólo lectura. Con la opción -t especificamos el sistema de archivos, hay distintos: fsck(ext2 o e2fsck, ext3, ext4, xfs), reiserfsck y dosfsck.

Examen y corrección del sistema de archivos

Usamos **debugfs que es** un **depurador interactivo de sistemas ext2** y **ext3**, con este es posible realizar tareas a bajo nivel como cambiar propiedades de inodos, borrar archivos, crear enlaces, etc. Se usa en casos donde fsck no soluciona el problema.

Otros comandos importantes son:

- **dumpe2fs**: muestra información de grupo de bloques y superbloques.
- **tune2fs**: configura parámetros ajustable de ext2.

Para **xfs** usamos:

- **xfs_metadump**: extrae todos los datos referentes al sistema de archivos en si.
- **xfs_info**: muestra características y otra información estadística sobre xfs.

Análisis de espacio en disco

Hay dos comandos principales:

- **df**: muestra el espacio ocupado y disponible en cada dispositivo en KB, con la opción -h usa medidas como MB y GB y con la opción -T muestra el tipo de sistema de archivo.
- **du**: muestra el espacio ocupado por archivos y/o directorios, con -s puede indicarse un directorio específico. Con la opción -h se usas medidas más legibles.

3)Control del montaje y desmontaje de los sistemas de archivos

Todos los sistemas de archivos creados en la instalación de Linux se montan automáticamente en el inicio del sistema. La información se almacena en /etc/fstab.

Fstab

Es necesario que exista una entrada para cada sistema de archivos en /etc/fstab, en este archivo se determinan las particiones, puntos de montaje, etc. Cada línea corresponde a un punto de montaje y contiene:

- **Partición del dispositivo.**
- **Punto de montaje.**
- **Tipo de sistema de archivos.**
- **Opciones de montaje:**
    - **rw**: lectura y escritura.
    - **ro**: sólo lectura.
    - **noauto**: no montar automáticamente.
    - **users**: podrá montarse y desmontarse por usuarios comunes.
    - **owner**: los permisos del dispositivo se adecuarán al usuario que lo montó.
    - **usrquota**: activa el uso de cuotas a usuario.
    - **grpquota**: activa el uso de cuotas a grupos.
    - **remount**: vuelve a montar el dispositivo.
- Dump: determina si el comando dump debe considerar el dispositivo, 0 ausente y 1 funcionando.
- Fsck: determina el orden del chequeo, para raíz 1 y ausente 0.

Montaje manual de sistemas de archivos

El comando es **mount**, si no se usan argumentos se muestran los dispositivos montados y detalles de ellos. Para montar un sistema de archivos que ya consta en /etc/fstab basta con informar al comando mount la localización del punto de montaje. Si usamos la opción **-a** se **montan todos** los dispositivos de **fstab excepto** los marcados con **noauto**.

Para **desmontar** usamos **umount punto_de_montaje**.

4)Administrar cuotas de disco

Es posible limitar la cantidad de espacio en disco a un determinado usuario o grupo. Para activar el control de cuotas es necesario que el kernel soporte esta función e incluya la opción usrquta o grpquota en el archivo /etc/fstab.

Análisis y control de cuotas

Para **generar la tabla de estadísticas de uso** de disco usamos **quotacheck -a**. Para **crear configuraciones** de cuota usamos **edquota**, con **-u** modificamos para el **usuario** especificado **y** con **-g** para el **grupo**.

La configuración de cada partición se almacena en la raíz del mismo y son aquota.user aquota.group.

Para que las cuotas pasen a **monitorearse y controlarse usamos quota -a**.

Es posible **avisar** al **usuario** cuando ha alcanzado su límite con **edquota -ta**.

El usuario root puede **generar informes de cuota** con **repquota -a**.

5) Controlar permisos y propiedades de los archivos

En sistemas Unix hay un sistema de permisos de archivos y directorios, existen tres niveles de permisos: dueño del archivo (u), grupo dueño del archivo (g) y otros (o).

Los permisos los podemos ver con ls -l. La primera letra indica que **tipo** de **archivo** es:

- **d:** directorio.
- **l**: enlace simbólico.
- **c**: dispositivo especial de caracteres.
- **p**: canal fifo.
- **s**: socket.
- **~**: archivo convencional.

Las demás letras se dividen en grupos de tres determinando los permisos.

Modificación de permisos

Para modificarlos usamos **chmod**, ejemplo: chmod g=r,o-r texto.txt (a otros le quitamos el permiso de lectura). Otro: chmod g+w texto.txt le añadimos permiso de escritura al grupo.

En directorios: r (permite acceder), w (permite crear archivos dentro) y x (permite listar el contenido de el).

Permisos octales

Manera más eficiente de manejar los permisos mediante formato octal, van en orden: usuario, grupo y otros.

Dígito	Lectura (valor 4)	Escritura (valor2)	Ejecución (valor1)
0	-	-	-
1	-	-	Si
2	-	Si	-
3	-	Si	Si
4	Si	-	-
5	Si	-	Si
6	Si	Si	-
7	Si	Si	Si

Ejemplo: chmod 664 texto.txt tendría u (4+2 LW), g (4+2 LW) y o(4 L).

Para cambiar los **permisos** de **todos** los archivos de un directorio usamos **-R.**

Umask

Es un **filtro de permisos** para la creación de archivos, cuando se crea un archivo o directorio se consulta esta máscara, por ejemplo cuando creamos una carpeta siempre es 777. Si usamos **umask sin argumentos se muestra la máscara actual** y para modificar le añadimos argumentos.

Suid y sgid

Todos los procesos se vinculan al usuario que los inicia, de esta manera el programa tendrá los mismos permisos de lectura y escritura que el usuario que lo ejecutó, pero para algunos programas o tareas se necesitan permisos de root.

Para solucionar esto existe un permiso especial **suid represendo** con la **letra s**, si un ejecutable lo tiene se **ejecutará con los permisos del dueño y no con los de quien lo ejecutó**. Para incluir este permiso, ejemplo: **chmod u+s programa**.

Sgid es **lo mismo** pero **para directorios**, en un directorio con permiso sgid todos los archivos creados pertenecerán al grupo del referido directorio (útil cuando muchos usuarios trabajan en un mismo grupo).

Al habilitar los permisos **suid** y **sgid aparece** la **letra s** en vez de x en los permisos, **si** tiene también **permiso de ejecución** aparecerá en minúsculas s **y si no S**.

Permiso sticky

Si compartimos directorios cualquier usuario puede borrar, pero con **sticky no** se **pueden borrar archivos no creados por el usuario mismo** (ejemplo /temp).

Los permisos con sticky aparecen con la letra t (en los permisos para otros usuarios) pero si sólo existe ese permiso aparecerá con T.

Modificar dueños y grupos de archivos

Para **modificar** el **dueño** usamos **chown usuario archivo/directorio** y para **grupos** igual pero con **chgrp**. Ejemplos:

chown luciano texto.txt

chgrp users texto.txt

chown luciano.users texto.txt (modifica los 2 a la vez).

Si añadimos la opción -R se modificaran todos los demás archivos repetidamente.

6) Crear y modificar enlaces simbólicos y duros

Son archivos especiales que sirven de atajos a otros archivos, existen dos tipos:

- **Físicos** (hard links): es uno de los varios nombres que puede tener un inodo en el sistema de archivos. Para **crearlos** usamos **ln archivo1 archivo2**. Podemos **listar** los enlaces con **ls -i**. **No** es posible crearlos para **directorios** y **sólo** se pueden crear para **archivos dentro** de un **mismo sistema de archivos**.
- **Simbólicos** (soft links): **pueden** apuntar **a cualquier objetivo**. Para **crear** uno usamos: **ln -l archivo1 archivo2**. Podemos **listarlos** con **ls -l** y **fijarnos** en la **letra l al inicio** y la fecha a que archivo apunta.

7) Encontrar archivos de sistema y conocer su localización

Todo archivo tiene una localización adecuada con respecto al estándar FHS.

FHS

Filesystem Hierarchy Standar, es el estándar de localización adoptado por casi todas las distros. Cada directorio sirve a un propósito

Directorios que residen obligatoriamente en la partición raíz.

Directorio	Finalidad
/bin y /sbin	Contienen los programas para cargar el so y comandos especiales.
/etc	Archivos de configuración específicos de la máquina.
/lib	Bibliotecas compartidas por /bin, /sbin y módulos del kernel.
/mnt y /media	Puntos de montaje.
/proc y /sys	Directorios con información de procesos y hardware.
/dev	Archivos de acceso a dispositivos y archivos especiales.

Directorios que pueden ser puntos de montaje

Directorio	Contenido
/boot	Kernel, mapas del sistema y cargadores de boot.
/home	Directorios de los usuarios.
/root	Directorio del usuario root.
/tmp	Archivos temporales.
/usr/local y /opt	Programas adicionales y blibliotecas.
/var	Datos de progrmas

Localización de archivos

Además de find podemos **buscar con locate**, es **más rápida que find** pues se **busca en** una **base de datos y no** en el **disco**. **Esta** base **debe actualizarse** regularmente **con** el comando **updatedb**.

El archivo **/etc/updatedb.conf** contiene que **directorios y sistemas de archivos debe ignorar la actualización** de la base de datos.

El comando **which** se usa para **devolver** el **camino completo al programa suministrado** buscando en PATH.

Con el comando **whereis** nos devuelve **los caminos al archivo ejecutable, código fuente y página manual.**

Tema 105: Shells, scripts y administración de datos

1) Personalizar y trabajar en el entorno de shell
Es la interfaz de interacción entre el pc y el entorno de programación, hay varias: bash, csh o zsh. El más usado es bash.

Variables:
Se usan para suministrar información útil y necesaria para programas y usuarios, se define de forma **nombre=valor**. **No** puede haber **espacios** y pueden ser:

- **Globales**: son todas aquellas accesibles por todos los procesos, como:
  - **PATH**: define directorios de programas.
  - **HOME**: define el directorio personal del usuario
  - **SHELL**: determina el shell estándar del usuario.
  - Los nombres de las **globales** son en **mayúsculas** y pueden **listarse** con el comando **env**, para listar **todas las variables se usa set.** Las variables **globales se definen en el login** y se encuentran en /etc/profile o para un usuario específico en ~/.bash_profile.

- **Locales**: son **sólo** accesibles **para** la **sesión actual de shell**. **Puden definirse en scripts o en la propia línea de comando**. Para dejar una variable accesible para las sesiones a partir de la actual usamos **export NOMBREVAR='contenido variable'**. Para **obtener el valor** de la variable usamos **echo $NOMBREVAR**, **para borrar** la variable usamos **unset NOMBREVAR**. Las variables **globales sólo pueden borrarse** para la **sesión actual**.

Funciones
Puden escribirse en línea de comando, en scripts o en archivos de configuración del bash. Ejemplo:

```
function pinfo () {
echo "localización de $1:"
which $1
echo "Procesos referentes a $1:"
ps x | grep $1
}
```

Para que **funcione** en **futuras sesiones de bash, la incluimos en ~/-bashrc**.

Archivo de configuración del bash
Hay 2 maneras de activar el bash:

- **Después del login de usuario** (shell interactivo): primero lee /etc/profile, luego ~/.bash_profile, ~/.bash_login, y ~/.profile y ejecutarálas instrucciones en ese orden. Al terminar la sesión ejecutará las instrucciones de ~/.bash_logout.
- **A partir de la sesión iniciada** (shell no interactivo): como los scripts ??

Si usamos shell interactivo pero no en sesión de login se ejecutará /etc/bash.bashrc y ~/.bashrc.

2)Editar y escribir scripts simples

Son archivos que actúan como programas, son interpretados.

Definición del interpretador

La primera línea del script identifica al interpretador con los caracteres **#!** (shebang), para instrucciones de bash sería #!/bin/bash. Para que pueda ser ejecutado debe de tener permisos, para ello chmod +x script.sh.

Variables especiales:

Los argumentos se devuelven con $x:

- **$***: Todos los valores pasados con argumentos.
- **$#**: El nº de argumentos.
- **$0**: El nombre del script.
- **$n**: El valor del argumento en la posición n.
- **$!**: PID del último programa ejecutado.
- **$$**: PID del shell actual.
- **$?**: Código de salida del último comando.

Para solicitar valores al usuario usamos read, eje: echo “dime edad” read EDAD, el valor se almacena en EDAD, si no se pone variable se usa REPLY.

If then else

if ejecuta un comando o lista de comando, test evalúa si la afirmación es verdadera o falsa, ejemplo:

```
if [ -x /bin/bash ] ; then
        echo “ok”
else
        echo “no ok”
fi
```

```
if test -x /bin/bash ; then
        echo “ok”
fi
```

Opciones de test para archivos y directorios:

- **-d camino**: true si el camino existe y es directorio.
- **-c camino**: true si existe el camino.
- **-f camino**: true si existe y es un archivo.
- **-L camino**: true si existe y es un enlace simbólico.
- **-r camino**: true si existe y puede leerse.
- **-s camino**: true si existe y si su tamaño es superior a 0.
- **-w camino**: true si existe y puede escribirse.
- **-x camino**: true si existe y es ejecutable.
- **camino1 -ot camino2**: true si camino1 es diferente a camino2.

Opciones de evaluación de test para texto:

- **-n texto**: true si el tamaño del texto es distinto a 0.
- **-z texto**: true si el tamaño de texto es 0.
- **texto1 == texto2**: true si texto1 es igual a texto2.
- **texto1 !=texto2**: true si texto1 es distinto a texto2.

Operaciones de evaluación de text para números:

- **num1 -lt num2**: true si numº < num2.
- **num1 -gt num2**: true si num1 > num2.
- **num1 -le num2**: true si num1 <= num2.
- **num1 -ge num2**: true si num1 >= num2.
- **num1 -eq num2**: true si num1 = num2.
- **num1 -ne num2**: true si num1 es diferente a num2.

Case:

La instrucción case proseguirá si un ítem indicado se encuentra en la lista de ítems:

```
case 3 in (1|2|3|4|5)
        echo “Numero 3 encontrado en la lista,”;
esac
```

Sustitución de comandos

Para mostrar o almacenar la salida de un comando, el mismo se coloca entre comillas simbles invertidas ` o entre $(), ejemplos:

```
TRESLINEAS=`cat -n3 /etc/inputrc`
echo “Las tres primeras líneas de /etc/inputrc:”
echo $TRESLINEAS
```

o

```
TRESLINEAS=$(cat -n3 /etc/inputrc)
echo “Las tres primeras líneas de /etc/inputrc:”
echo $TRESLINEAS
```

Operaciones matemáticas

Para operar con números enteros usamos expr:

```
SUMA=`expr $VALOR1 + $VALOR2`
```

o

```
SUMA=$((expr $VALOR1 + $VALOR2))
```

Instrucciones de bucle

For

Ejecuta una o más acciones para cada elemento de una lista, ejemplo con seq:

```
for i in $(seq 5); do
        echo “Bajando foto_$i.jpg”;
        echo wget http://www.marchicosita.com/foto_$i.jpg;
done
```

Realizaría la acción 5 veces, seq aumenta de uno en uno.

Until

Ejecuta una acción en loop hasta que la afirmación sea verdadera, ej:

```
LENTEXTO=$(wc -l texto_simple)
        until [ ${LENTEXTO%% *} -eq 10}; do
        echo “Una línea más” >> texto_simple
        LENTEXTO=$(wc -l texto_simple)
done
```

Añadirá una línea más y contará el nº de líneas hasta que sea 10 y pare.

While

Ejecuta una instrucción hasta que una afirmación no sea verdadera

```
LENTEXTO=$(wc -l texto_simple)
        while [ ${LENTEXTO%% *} -lt 10}; do
        echo “Una línea más” >> texto_simple
        LENTEXTO=$(wc -l texto_simple)
done
```

Añadirá una línea más y contará el nº de líneas mientras no haya 20 líneas.

Local, propiedad y permiso

Es importante que sea ejecutable y que esté en un directorio incluido en la variable PATH, el derecho de escritura debe quitarse de todos excepto del dueño y no es recomendable activar el bit de SUID.

3)Administración de datos SQL

SQL (Structured Query Language) o lenguaje de consulta estructurada, es el estándar para la realización de consultas.

Interacturar con los datos

La más básica comunicación se realiza por línea de comandos. Cada tecnología tiene su propia herramienta MySQL (mysql) y para Postgresql (psql).

Las bases de datos se organizan en tablas que contienen columnas y filas.

Inserción de datos

Con el comando **INSERT**, ejemplo:

INSERT INTO cliente (nombre, correo) VALUES ('Surmano', 'ayoma@querica.com');

Consultas de datos

Con el comando **SELECT**, ejemplos:

- SELECT * FROM cliente; Todas las filas.
- SELECT id, nombre FROM cliente; Todas las filas pero sólo id y nombre.
- SELECT nombre FROM cliente WHERE id=2; Todas donde el id sea 2.
- SELECT id, nombre FROM cliente ORDER BY id DESC; Todas las filas pero sólo id y nombre y ordenadas por id en orden descendiente.
- SELECT nombre, SUM(precio) FROM item GROUP BY nombre; Todas las filas de item agrupadas por nombre y sumando el precio. Desaparecerán los duplicados por nombre y sus precios se sumarán.

Modificación y borrado

Modificar con el comando **UPDATE**, ejemplo:

UPDATE cliente SET telefono = '999999999' WHERE id=2;

Podemos modificar más de una columna para cada registro.

Para **borrar** con el comando **DELETE**, ejemplo:

DELETE FROM cliente WHERE id=2;

Relación de tablas

Una de las ventajas de la DB es poder relacionar las tablas, con el comando **INNER JOIN** podemos relacionarlas.

SELECT cliente.nombre, direccion.prov FROM cliente INNER JOIN direccion ON direccion.id_cliente=cliente.id;

Tema 106: Interfaces de usuario y escritorio.

1) Instalar y configurar el X11

El entorno de ventanas se llama X11 pero es conocido como X. Es posible hacer login en una sesión del X11 por red o mostrar una ventana de un programa en otro pc.

Compatibilidad de hardware:
La configuración del X11 se realiza **automáticamente durante la instalación** de la distro. Podemos ver la compatibilidad de hardware en http://www.x.org/wiki/Projects/Drivers. Si un dispositivo **no es totalmente compatible podemos** usarlo en **modo VESA** que casi todos los dispositivos son compatibles.

Instalación X11
Lo común es que se instale durante la instalación, si no puede hacerse desde apt-get o yum.

Configuración del X11R6
Configurar manualmente significa **editar /etc/X11/xorg.conf** donde se encuentra toda la configuración, para generar un archivo básico se usa **X -configure** y el servidor X probará el driver y guardará el resultado en xorg.conf.new en /.

Secciones de xorg.conf
Se divide en secciones en el formato:

```
Section "nombre seccion"
        Item_1 "valor item 1"
        Item_2 "valor item 2"
EndSection
```

La mayoría de las secciones ya no son necesarias porque la configuración se realiza automáticamente. Secciones:

- Files: caminos para archivos necesarios como FontPath.
- ServerFlags: opciones globales en formato Option "Nombre" "Valor".
- Module: carga dinámica de módulos, Load "nombre_modulo".
- InputDevice: dispositivos de entrada, con Identifier y Driver.
- Device: dispositivo de vídeo y puede tener varias secciones.
- Monitor: con varias secciones.
- Screen: agrega el dispositivo y el monitor.
- Display: subsección de Screen.
- ServerLayout: agrega Screen e InputDevice.

Fuentes
X11 suministra las fuentes usadas por las aplicaciones, existen **2 sistemas**:

- **Core**: las fuentes se manipulan en el servidor.
- **Xft**: es más avanzado y permite fuentes Type1, OpenType y usar anti-alising.

Para **instalar fuentes Xft** basta con **copiarlas a /usr/share/fonts/* o a ~/fonts/ y** luego **actualizar** la **caché con fc-cache**. El comportamiento de las funciones de Xft puede modificarse en /etc/fonts/fonts.conf.

Servidor de fuentes Xfs: en entornos de red es posible tener un único servidor de fuentes que las compartirá con las demás máquinas. El **puerto** estándar es el **7100** para configurar una máquina para que las use hace falta configurar el **FontPath "tcp/ipdelservidor:7100"**

Variable DISPLAY
Permite que las ventanas de las **aplicaciones se muestren en un servidor X remoto**. A partir de esta variable de entorno DISPLAY el servidor de ventanas idenfitica donde debe mostrarse.
La variable se compone de :nombre_o_ipmaquina.display de la máquina. En una máquina con un sólo display la variable será :0.0.
Para abrir una ventana remota primero hay que redefinir la variable display en la máquina remota, ej: **export DISPLAY=192.168.1.3:0.0** Con esto cualquier programa ejecutado después del cambio se enviaría a esa máquina pero no se mostrará hasta que esta lo permita, para ello se escribiría **xhost +192.168.1.1**

2)Configuración del gestor de login gráfico

Por defecto X11 se incia tras terminar de arrancar Linux apareciendo la ventana de login.

Gestor de pantalla
El programa encargado de inciiar sesión e identificar al usuario es display manager o gestor de pantalla, existen 3 principales:

- **xdm**: estándar del X, archivos de config en /etc/X11/xdm/*. También permite el login por red usando el protocolo XDMCP (desactivado por defecto) configurable en /etc/X11/xdm/Xaccess.
- **gdm**: Gnome, archivos de config en /etc/gdm(.
- **kdm**: KDE. , archivos de config en /usr/share/config/kdm/*.

3)Accesibilidad

Podemos activarla en Sistema | Preferenias | Tecnologías de asistencia*.

Orca: Es un lector de pantalla que nos irá leyendo el texto seleccionao cuando se hace click sobre algo.
GOK: si no es posible usar teclado se realizará mediante el ratón.
Preferencias del mouse: si se tiene una dificultad motora y no se puede manejar bien se puede personalizar el comportamiento del ratón.

Tema 107:

1) Cuentas de usuario:
Usamos **#useradd** para añadir usuarios. También podemos usar **#adduser** para facilitar la creación de usuarios ya que se le puede definir valores de creación en /etc/adduser.conf
Parámetros de **useradd**:

- **-c** comentario: suele ser el nombre entero.
- **-d** directorio: ruta del home
- **-g** grupo grupo inicial ya existente (GID)
- **-G** grupo1, grupo2 para asignar más grupos.
- **-u** UID: el user id del usuario.
- **-s** shell: shell estandar para el usuario. Luego puede modificarse con #chsh
- **-p** "contraseña": la contraseña entre ". Podemos cambiar la clave con #passwd usuario, si se usa sin argumento se modifica la del usuario actual.
- **-e** fecha: fecha hasta la que la cuenta está activa.
- **-k /etc/skel**: copia el home modelo de /etc/skel para poder personalizar el home.
- **-m:** crea el directorio personal si no existe.

Con chfn se puede cambiar la descripción del usuario.
Los usuarios pueden usar también passwd, chsh y chfn para cambiar sus datos.
Con **#userdel** usuario borramos el usuario, si le añadimos el parámetro –r nos aseguramos de que se borre su home.
La información de los usuarios del sistema se almacena en /etc/passwd en formato:
Nombre de login**:**contraseña se almacena en /etc/shadow**:**UID**:**GID**:**Descripcion**:**home**:**shell

Para editar **/etc/passwd** se recomienda usar el comando **#vipw** ya que este bloquea el archivo para la edición y evita la corrupción del mismo.
Cualquier usuario puede leer /etc/passwd para evitar esto las contraseñas se almacenan en /etc/shadow sólo accesible desde root.
Si se tiene una versión antigua y se almacenan en /etc/passwd se puede realizar la migración a /etc/shadow con el comando **pwconv** o **pwuconv** para lo contrario.

El formato de **/etc/shadow** es:

- Nombre de login**:**
- pass encriptada en 13 caracteres (en blanco no necesita pass con * está bloqueada)**:**
- Nº de días desde el 01/01/1970 que se cambio la pass**:**Nº de días necesarios hasta que la pass pueda modificarse (0 no necesita tiempo)**:**
- Nº de días después del cual debe modificar la contraseña(por defecto 99999 o 274 años):
- Nº de días para informar al usuario de la caducidad de la pass**:**
- Nº de días después de la expiración de la cuenta hasta que se bloquee**:**
- Nº de días a partir de 01/01/1970 desde que se bloqueo la cuenta**:**
- Campo reservado.

Con **#chage** se puede modificar la validez de la contraseña, opciones:

- -m días: mínimo de días hasta que el user pueda cambiar la contraseña modificada.
- -M días: nº máximo de días en el cual la contraseña permanecerá válida.
- -d días: nº de días transcurridos entre que se cambió la pass y 01/01/1970.
- -E días: nº de días transcurridos a partir del cúal la cuenta está desactivada desde 01/01/1970.
- -I días: nº de días de tolerancia después de que la cuenta se desactive para que se bloquee la cuenta.
- -W días: nº de dias antes de que caduque la contraseña en la que se mostrará un aviso de caducidad.

Los usuarios normales pueden usar **chage -l usuario** que muestra las restricciones del usuario. También pueden usar usermod para modificar la cuenta:

- -c descripcción: descripción del user.
- -d directorio: modifica el home, si añadimos el argumento -m, mueve el contenido anterior.
- -e valor: plazo de validez de la cuenta en formato dd/mm/aaaa.
- -f valor: nº de días después de que la contraseña expiró hasta que se bloquee. Con -1 cancelamos esta opción.
- -g grupo: grupo efectuvo del usuario.
- -G grupo1,grupo2: grupos adicionales del usuario.
- -l nombre: nombre del login.
- -p contraseña: contraseña.
- -u UID: nº de identificación del usuario UID.
- -s shell: shell estandar del usuario.
- -L bloquea la cuenta del usuario con ! delante de la contraseña. Como alternativa podemos cambiarle el shell estandar por un script que informe del bloqueo.
- -U: desbloquea la cuenta de usuario borrando el !.

2)Grupos de usuarios:

Usamos **#groupadd** , ejemplo: #groupadd estudio_c. Con el parámetro -g podemos especificar el GID. Para eliminar el grupo usamos **#groupdel** nombre.

Para añadir/eliminar usuarios a un grupo usamos **#gpasswd** con estos parámetros:

- gpasswd grupo: crea una contraseña para el grupo.
- -r grupo: borra la contraseña del grupo.
- -a usuario grupo: asocia un usuario a un grupo.
- -d usuario grupo: excluye el usuario del grupo.
- -A usuario grupo: convierte un usuario en administrador del grupo.

Un usuario puede permanecer en más de un grupo. Para mostrar los grupos al que pertenece un usuario usamos **groups usuario**. Si no se indica el usuario, este muestra los grupo del usuario actual.
El comando **id** muestra los grupos de usuario con el GID correspondiente.

Con newgrp se usa para modificar el grupo efectivo del usuario en una nueva sesión de login.

La información de los grupos se almacena en **/etc/group** en el siguiente formato:

- Nombre del grupo**:**
- Contraseña del grupo (x si se usa **/etc/gshadow**)**:**
- GID**:**
- Miembros del grupo separados por coma.

Para editar /etc/group se recomienda el comando **vigr** que bloquea evitando corrupción. Si usamos **vigr -s** se abrirá /etc/gshadow para edición.
El comando **grpconv** convierte contraseñas antiguas (/etc/group) al nuevo (/etc/gshadow) y **grpunconv** para lo contrario.

Con el comando **#groupmod** se puede también modificar grupos con estas opciones:

- -g GID: modifica el GID.
- -n nombre: modifica el nombre del grupo.

3) Automatizar y programar tareas:

Hay 2 sistemas:

- **at:** Se usa para programar una ejecución única. Los usuarios normales pueden usarlo si constan en el archivo **/etc/at.allow**, si este no existe se consultará **/etc/at.deny** y se bloqueará a los que consten en él, pero si ninguno de los dos archivos existe sólo root puede usar at. Su sintaxis es: **#at cuando comando.** Cuando puede significar now(ahora) o midnight(medianoche), otras opciones pueden consultarse en /usr/share/doc/at/timespec. Para verificar si hay tareas pendientes usamos **at -l** o **atq**.

- **Cron**: Se usa para ejecutar una tarea en intervalos. Cada min el daemon **crond** lee las **crontabs** (tablas de tareas) para ver las tareas. El crontab general se almacena en **/etc/crontab** y no se debe editar directamente si no a través del comando **crontab**:

    - -l usuario: muestras las tareas programadas por el usuario.
    - -e usuario: edita el contab en el editor estandar.
    - -d usuario: borra el crontab del usuario.

    Si no se suministra el usuario, se interpretará que es el usuario actual.
    Formato :
    0-59(Min) 0-23(Hora) 0-31(Dia) 1-12(Mes) 0-6(Día semana) comando. Si usamos "-" entre dos nº delimita un periodo de ejecución, si usamos el carácter "*" (sin nada) se ejecutará a cada momento de ese intervalo. Si usamos "/" establece un paso de ejecución (*/4 cada 4 horas).
    Ejemplo: * */4 5,6 1-5 /usr/local/bin/script_backup ,es decir, ejecuta ese script cada 4 horas de martes a sábado los meses Mayo y Junio.
    Si la tarea produce alguna salida se enviará a la caja de entrada del usuario, pero modemos evitar esto redireccionando a /dev/null o a un archivo.
    Controlamos el uso de crontab mirando en los archivos /etc/cron.allow y /etc/cron.deny. Si no existe ninguno de estos todos los usuarios podrán programar tareas.

4) Localización e internacionalización:

Linux soporta diversos idiomas y husos horarios.

- **Huso horario**: va en relación a GMT dependiendo el país. Se recomienda usar en la bios GMT+0. Para informar al sistema del huso horario deseado usamos **tzselect** y después de seleccionar el deseado se crea el archivo **/etc/timezone** con los datos de la zona. También se crea **/etc/localtime** para las horas. Todos los archivos de huso horario se encuentran en /usr/share/zoneinfo.

- **Idioma y codificación**: La configuración básica de localización se realiza con la **variable de entorno LANG** y a partir de esta los programas predefinen su idioma. Esta variable está en **formato ab_CD** donde ab es el idioma y CD es la codificación (es.UTF-8). Para convertir textos entre diferentes codificaciones usamos: **iconv** -f iso-8859-1 -t utf-8 txtoriginal txt salida. Además de LANG se usan otras variables como:

    - LC_COLLATE: define el orden alfabético. (al hacer ls que muestre el orden).
    - LC_CTYPE: define como el sistema trata ciertos caracteres.
    - LC_MESSAGES: definición de idioma de los avisos emitidos por los programas.
    - LC_MONETARY: define la moneda y su formato.
    - LC_NUMERIC: formato numérico de los valores no monetarios.
    - LC_TIME: formato fecha y hora.
    - LC_PAPER: tamaño estandar del papel.
    - LC_ALL: superpone todas las demás variables.

- **Opciones de idioma en scripts**: en los scripts definimos la **variable LANG=C** para que el script no produzca resultados diferentes si la localización fuera distinta a aquella donde se escribió.

Tema 108:

1) Mantenimiento de la fecha y hora del sistema:
Si está erroneo pueden corromperse algunos servicios y mantenimientos programados.

Relojes: el reloj de la BIOS sólo se lee durante el boot, luego usa uno separado de este.
Con el comando date se muestra la fecha y la hora, si le añadimos el parámetro -u la muestra en formato UTC(GMT+0). Para modificar el reloj de la BIOS usamos hwclock:
-w actualiza el reloj de la BIOS con la hora del sistema.
-s actualiza la hora del sistema con la de la BIOS.
-u indica que se usa UTC.

NTP: Network Time Protocol: un pc en red puede actualizar la hora comparándola con la hora de otro pc con reloj más exacto. Para que el sistema use NTP el archivo **/etc/ntp.conf** debe estar configurado y el **servicio ntpd activo**. Este servicio usa UDP con el puerto 123. Ejemplo de **ntpd.conf**:

server es.pool.ntp.org
server 0.pool.ntp.org
driftfile /etc/ntp.drift (almacena las estadisticas de error)

Lentamente va ajustando a hora para evitar cambios bruscos, pero podemos **forzar** el cambio con **#ntpdate servidor**.

2) Configurar y recurrir a los archivos log:

Guardan registro de las operaciones del pc. La mayoría se encuentran en **/var/log** y son controlados por el **servicio syslog**.
El **syslog se configura por el archivo /etc/syslog.conf**, cada regla se separa en selector y acción separados por espacios o tabulaciones:

- **selector**: se divide en:
  - **facility:** que identifica el mensaje de origen (auth, authpriv, cron, daemon, daemon, fpt, kern, lpr, mail, news, syslog, user, uucp y local0 hasta el 7).
  - **priority:** que varían de menos urgentes a más urgentes (debug, info, notice, warning, err, crit, alert y emerg) separadas por un punto. Con none se ignora la urgencia.

  Los dos pueden tener caracteres modificables como * que indica que vale para cualquier facility o priority, dependiendo donde esté. En la misma regla pueden especifigarse más de una facility separadas por comas. El signo = otorga exclusividad a facility/priority que lo sucede, y en contrapartida el signo ! Para lo contrario. El signo ; puede usarse para separar más de un selector para la misma acción.

- **Action**: determina el destino dependiendo del mensaje. Normalmente los mensajes se envian a /var/log pero podemos direccionarlo a pipes, consolas, host, etc.

Es posible hacer que **syslog registre mensajes de otras máquinas de la red**, para ello iniciamos en el servidor el programa **syslogd -r**.
Como los log son ampliados continuamente es recomendable **mover los mensajes antiguos** a otro lado, esto se hace mediante el comando **logrotate** (se usa normalmente con cron). Su configuración es en **/etc/logrotate.conf**
Podemos también **crear entradas manuales de log** con el comando **logger** comando. Si añadimos -p podemos indicar el facility/priority.

3) Fundamentos de MTA: Mail Transfer Agent

El programa que controla el **envio/recepción de mails se denomina MTA**. Existen varias opciones: **sendmail, postfix, qmail y exim**. Funciona como servicio del sistema a través del **puerto 25** responsable del protocolo **SMTP**.
El **direccionamiento** puede definirse en **/etc/aliases**. Si modificamos este, tenemos que ejecutar **newaliases** para que se apliquen. También es posible reenviar editando el archivo .forward del home.
Para **mostar la cola** de correo electronico y el **estado de transferencia** usamos el comando **mailq**.

4) Configuración de impresoras e impresión.

El programa responsable de la impresión se denomina **CUPS** y es compatible con herramientas del sistema de impresión antiguo lpd. La configuración de CUPS se puede hacer modoficando sus archivos, por línea de comandos y por web (es la más recomendada).

Para acceder por web usamos **http://localhost:631**, para que este funcione tiene que estar activo el servidor de impresión **/usr/sbin/cupsd** que inicia en el boot en la mayoría de distros.

Archivos de configuración de CUPS (/etc/cups/):

- clases.conf: define las clases de impresoras locales.
- cupsd.conf configuraciones del daemon cupsd.
- mime.convs: define los filtros disponibles para conversión de formatos de archivos.
- printers.conf: define los tipos de archivos conocidos.
- lpoptions: configuraciones específicas para cada impresora.

Para la **administración** de impresoras usamos **lpadmin**:

- -c clase: agrega la impresora indicada a una clase, si no existe se creará.
- -m modelo: especifica el driver estandar de la impresora (generalmente archivo PPD).
- -r clase: elimina la impresora elegida.
- -v: dispositivo: indica el interface que usará la impresora.
- -D info: descripción textual de la impresora.
- -E: autoriza la impresora a que reciba trabajos.
- -L localización: descripción textual para la localización de la impresora.
- -P archivo PPD: especifica el driver de la impresora.
- -x nombre: se elimina la impresora por nombre.

Para ver datos por comandos **lpingo**:

- -v: muestra la lista de dispositivos de impresión y sus protocolos.
- -m: muestra los modelos de impresoras disponibles.

Para **modificar opciones de impresión** tenemos que ejecutar **lpoptions,** para **ver el estado de las impresoras lpstat -a** y para **ver la cola de impresión lpq** (-a todas las colas del sistema y -P las de la máquina especificada) **. Esta cola se almacena en /var/spool/cups/**.

Para **imprimir archivos** usamos el comando **lpr**:

- -Pxxx: envía el archivo a la cola xxx.
- -#x: imprime el documento x veces.
- -s: no copia a la cola de impresión, si no que crea un enlace simbólico.

Para **cancelar un trabajo de impresión** usamos **lprm nº** (si no se indica nº se cancelará el último mandado) y para **cancelar todo lprm -a** o lprm -.

Una impresora también puede configurarse para ser compartida por red desde CUPS y desde Samba.

Tema 109: Fundamentos de red:

1) Fundamentos de los protocolos de Internet

Dirección IP está en formato X.X.X.X que es la expresión en decimal de una dirección de red binaria. Ej: 192.168.1.1 = 11000000.10101000.00000001.00000001. Cada interfaz de red debe de tener una dirección IP única.

Clases de redes: existen rangos específicos que no deben de aplicarse a las LAN:

- **Clase A**: 1.0.0.0 a 127.0.0.0. 8 bits de red y 24 de hosts.
- **Clase B**: 128.0.0.0 a 191.255.0.0. 16 bits de red y 16 de hosts.
- **Clase C**: 192.0.0.0 a 223.255.255.0. 24 bits de red y 8 de hosts.

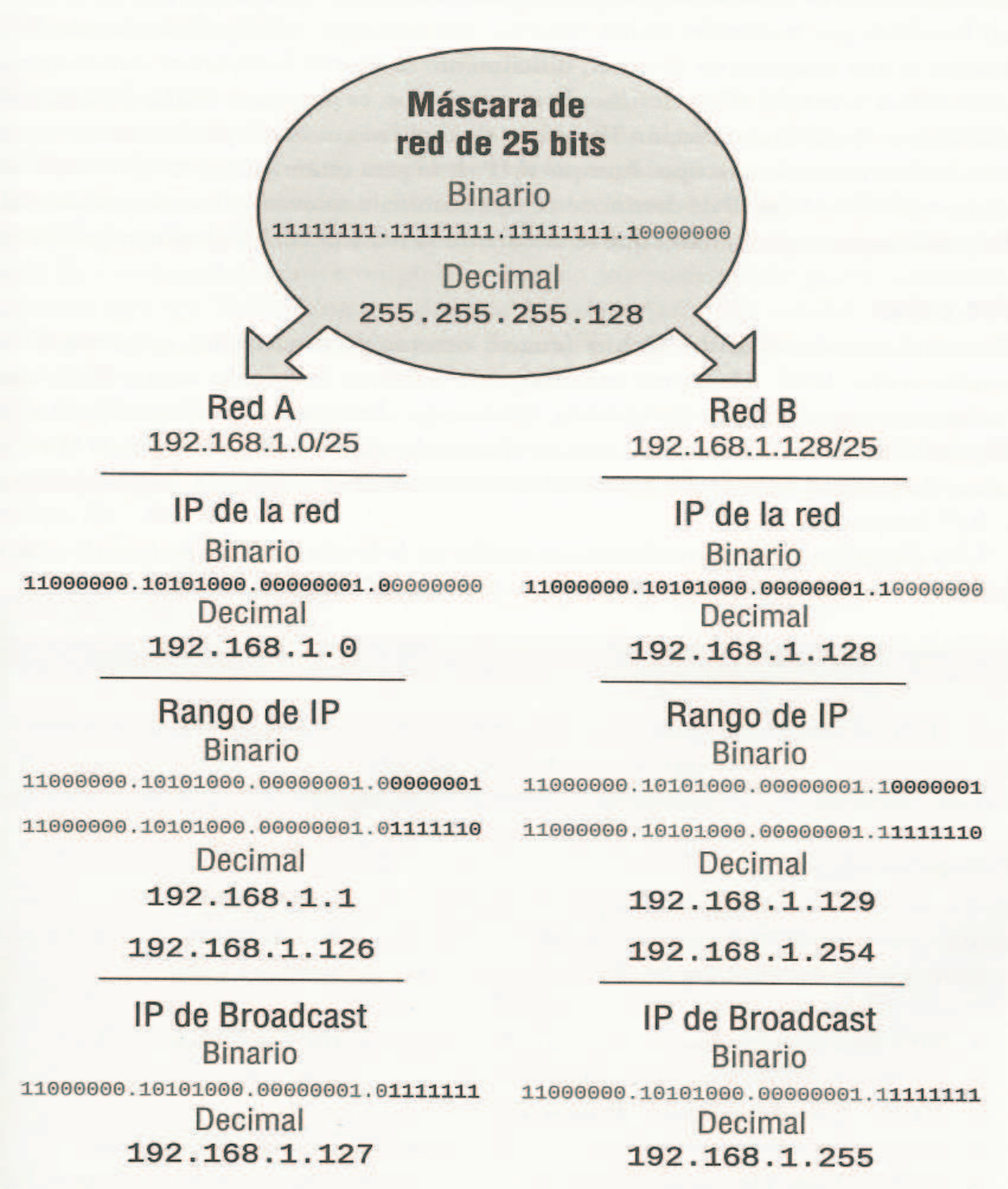
Para que los datos puedan encaminarse correctamente por la red, la interfaz de red necesita conocer:

- su IP.
- IP de destino.
- a que red pertenecen, para ello necesitan conocer su máscara de red. Para calcularla lo pasamos a binario y si coinciden los bits de red es la misma.

Ip del interfaz	**AND** 00=0 01=0 10=0 11=1	**Máscara de la red**	**Igual a:**	**Ip de la red**
Ip de la red	**XOR** 00=0 01=1 10=1 11=0	**Máscara de la red**	**Igual a:**	**Ip de la Broadcast**

Una dirección puede venir en formato abreviado, ej: 192.168.1.129/25 lo que indica: ip=192.168.1.129, mask: 255.255.255.128 y broadcast: 192.168.1.255

Subredes: una red puede dividirse en varias redefiniendo su máscara de red. De esta manera **una red puede dividirse en redes menores (sin clase) denominadas CIDR**. Por ejemplo:
Red 192.168.1.0 y mask 255.255.255.0 puede dividirse en dos al activar el primer bit del cuarto octeto en la máscara. Por lo que los primeros 25 bits serían para red. Si usamos 26 bits dividiríamos al red en 4.



Ruta estándar: cuando el destino no está en la misma red (una máquina de internet), es necesario **establecer una ruta por donde se encaminarán los datos** de este tipo (puerta de enlace?).

Ipv4 es de 32 bits pero Ipv6 es de 128 bits, esta versión 6 permite michas más direcciones IP. Está formada por **ocho grupos de cuatro números hexadecimales**.

Protocolos de red: constituyen el lenguaje utilizado para la comunicación de dos máquinas:

- IP: protocolo base usado en TCP, UDP e ICMP para enrutamiento.
- **TCP** (transfer control protocol): protocolo de control de formato e integridad de los datos.
- **UDP** (user datagram protocol): igual que TCP pero los controles sufren intervención de la app.
- **ICMP** (internet control message protocol): permite la comunicación de enrutadores y host para indentificar y comprobar el estado de funcionamiento de la red.

Puertos TCP y UDP: los protocolos hacen posible la comunicación de los servicios de red, asignando un puerto específico a cada uno de estos. Es necesario que los pcs conectados respeten los nº de puertos correctos para cada servicio. IANA (internet assigned numbers authority) controla la lista oficial de puertos:

Puerto	Servicio
20	FTP (puerto de datos)
21	FTP
22	SSH
23	Telnet
25	SMTP
53	DNS
80	HTTP
110	POP3
119	NNTP (usenet)
139	Netbios
143	IMAP
161	SNMP
443	HTTPS
465	SMTPS
993	IMAPS
995	POP3S

En Linux la **lista de servicios y puertos se almacena en /etc/services**, existe un másximo de **65535 puertos**.

2) Configuración básica de red

Aunque los protocolos sean los mismos para los distintos SO hay distintas herramientas.
Archivos de configuración: todas las configuraciones quedan almacenadas en archivos de textos en el directorio /etc/:

- hostname: nombre de la máquina local.
- hosts: asigna ips a nombres, para acceder a máquinas más facilmente en redes pequeñas.
- nsswitch.conf: indica donde debe el sistema empezar a buscar recursos. Claves: files, nis y dns.
- Resolv.conf: donde se almacenan los DNS.

Configuración manual de la interfaz: a través de ifconfig:

- interfaz down: desactiva el interfaz (también ifdown interfaz).
- Interfaz direccionIP netmask Máscara up: asigna esa ip, esa máscara y activa el interfaz (también ifup interfaz, lo activa).
- Sin argumentos: muestra información de los interfaces de red.
- Interfaz: muestra información de ese interfaz.

Configuración de rutas, el comando route muestra y crea rutas. Route:

- -n: muestra la tabla de rutas. El cambo Flags muestra el estado de la ruta.
- Add -net Ip_red netmask máscara dev interfaz: crea una ruta para el interfaz determinado, para esa red y esa máscara.
- Add default gw PuertaDeEnlace: añade una ruta estándar a esa IP. Otra forma sería route add -net 0.0.0.0 dev Interfaz: que crea una ruta para cualquier red que no tenga ruta especificada.
- Del default gw PuertaDeEnlace: borra la ruta estandar por la máquina PuertaDeEnlace.

Con el comando ping puede usarse para comprobar el funcionamiento de la red. Por ejemplo: ping -c3 PuertaDeEnlace, este envia 3 paquetes a la puerta de enlace, si responde está correcto.

3) Soluciones para problemas simples de red

Inspección de la configuración:

- **ifconfig**: Verificamos la configuración del interfaz.
- **Ping:** para ver si hay conectividad.
- **Route**: para ver que las rutas estén correctas.
- **Host dominio**: para hacer consulta de dns para ver si estas funcionan. Podemos usar **dig** que **es más completo.**
- **Netstat**: para analizar el tráfico:
  - -n: sólo muestra Ips.
  - -t: muestra sólo TCP
  - -c: muestra continuamente las nuevas conexiones.
  - -i: muestra información de todas la redes activas y sus estadísticas.
  - -r: muestra la tabla de rutas del sistema.
- **Traceroute**: muestra las rutas recorridas por un paquete hasta su destino. Con -n mostramos sólo las ips.

Configuración de DNS del cliente:

- Hay que **verificar /etc/nsswitch.conf** y verificar que en el **apartado host esté puesto dns.**
- Verificamos el archivo **/etc/hosts** para verificar **que no tenga redirecciones erroneas**
- Verificamos el archivo **/etc/resolv.conf** para **verificar que tenga dns asignados**.

Tema 110: Seguridad:

Verificaciones de permiso:
Archivos con permiso SUID y SGID garantizan privilegios especiales a quienes lo ejecutan, un comando modificado que contenga ese permiso especial podrá dar acceso de root a un usuario común. Por lo tanto hay que monitorear que archivos tienen esos permisos, para ello usamos:
#find / -perm -4000 -or -perm -2000

Varios comandos deben de tener estos permisos especiales. **Para facilitar el chequeo podemos generar una lista detallada** (recomendable diariamente por cron):
#find / \(-perm -40000 -or -perm -2000 \) –exec ls –l '{}' \; > /var/log/setuid-$(date +%F)

Que generará un archivo con el nombre setuid-año-mes-dia que **podremos comprar con** el de otros días mediante el comando diff:
#diff /var/log/setuid-fecha_anterior /var/log/setuid-fecha_actual

Otra búsqueda a tener en cuenta es la de **archivos con permiso de escritura para todos** los usuarios excepto en **/dev** ya que podrían modificarse para ser viable invasiones o daños. Para buscar estos archivos ejecutamos:
#find / -path /dev -prune -perm -2 -not –type l

Para buscar archivos **sin dueño o sin grupo** usamos: **#find / \(-nouser -o -nogroup \)**

También hay que **revisar que las contraseñas** de los usuarios **estén en /etc/shadow**.
Retoque de contraseñas, con el comando **#passwd parámetros usuario/grupo**:

- -e: provoca la expiración inmediata de la contraseña.
- -d: borra la contraseña de la cuenta.
- -g: modifica la contraseña del grupo seleccionado. Seguido de: -r: borra la contraseña y con -R restringe el acceso a todos los usuarios.

Una cuenta también puede **desactivarse** con **passwd –l** y activarse con **passwd –u**, el estado de la cuenta con **passwd –S. Formato passwd –S**:

- Login
- P(tiene pass)/NP(no tiene)/L(cuenta bloqueada)
- Fecha del último cambio de contraseña.
- Límite mínimo de días de la contraseña.
- Límite máximo de días de la contraseña.
- Días de aviso.
- Límite de días de inactividad desde que la contraseña caduca hasta que se bloquee la cuenta.
- Los atributos de la contraseña se cambian con **#chage** (ver temas anteriores).

Acceso como root:

Para entrar como **root** se usa el comando **su**. **Si no se ponen argumentos**, esta nueva sesión **heredará** las configuraciones de entorno y variables de la sesión donde se ejecuta.
Si usamos **su –l o su -** se **creará una nueva sesión** totalmente **distinta.**

El comando **sudo permite** a un usuario o grupo **realizar tareas antes reservadas a root**, para ello estos usuarios tienen que permanecer al grupo admin o sudo.

Limitación de recursos:
Los usuarios pueden provocar lentitud o paradas del sistema por el uso de los recursos. Por ello **debemos de limitar la cantidad de recursos** máximos que pueden usar en el sistema, se hace a través de **#ulimit** y este actúa a nivel de sesión del Bash por lo que sólo afecta a los procesos activos en esa sesión. **Se ha de establecer** en cada recurso **un límite soft (-S**) que es el nivel de alerta y un **límite hard (-H**) que es el valor máximo.
Ulimit:

- -a muestra los límites actuales.
- -f: especifica el nº maximo de archivos que podrá haber en la sesión de shell, es la opción por defecto si no se le añade parámetros a ulimit.
- -u: número máximo de procesos disponibles para el usuario.
- -v: valor máximo de memoria virtual disponible para el shell.

Verificación de los puertos abiertos en el sistema
Usamos el programa **nmap** dirección, por ejemplo **#nmap localhost** y mostrará todos los puertos abiertos del equipo local, también podemos hacer esto con **netstat**.

También podemos descubrir los puertos abiertos y el SO de un objetivo:

nmap –o IP_LAN

Con **netstat –l** obtenemos los puertos esperando conexiones del sistema local.
Para **identificar que programa** y usuario **está usando un puerto** usamos **lsof**
Ej: **lsof –i máquina:puerto** -> lsof -1 127.0.0.1:22

2) Seguridad del host
Desactivación de servicios de red:
Puede haber **servicios de red que no usemos** y es **recomendable tener desactivados**. Para ello **finalizamos el servicio y ejecutamos chmod –x script** para **quitar**le el permiso de **ejecución**.
Para servicios activados que **no** queramos que **inicie por inetd (comentamos la línea con #**) y **xinetd (cambiamos disable a = yes)**, el archivo se encuentra en **/etc/inetd.conf y /etc/xinetd.conf**, aunque **pueden estar fragmentados** en subcarpetas dentro de **/etc/inetd.d/ y /etc/xinetd.d/**.
Normalmente los **servicios se interrumpen** por medio del propio script que los inicia pero poniendo el argumento **stop**, **ej: /etc/init.d/samba stop.**
Podemos **bloquear el acceso al sistema** de todos los usuario **excepto de root creando** el archivo **/etc/nologin**, al intentar entrar los **usuarios normales se les mostrará el contenido de ese archivo**.

TCP warppers:
Se usa para **controlar el acceso a los servicios** disponibles, este control se realiza por **reglas** en **/etc/hosts.allow y /etc/hosts.deny** que contiene direcciones remotas que podrán o no acceder a la máquina local. Primero se mira .allow y luego deny, pero si no consta en ninguno se autorizará. Se escriben en formato **servicio:host:comando**.

- Servicio: es uno o más nombre de daemons de servicio
- Host: es una o varias direcciones, puede ser dominio, ip completa o incompleta con *.
- Comando: es un comando opcional que hará que se cumpla la regla.

En host y servicio pueden usarse instrucciones especiales como: ALL, LOCAL, KNOW, UNKNOWN y PARANOID.
Ejemplo de regla para autorizard todos los servicios para 192.168.1.0 excepto para .20:

- En /etc/hosts.allow: ALL: 192.168.1.* EXCEPT 192.168.1.20
- En /etc/host.deny: ALL: ALL (bloquea todo los servicios a toda dirección que no esté en .allow).

3) Protección de datos con encriptación:
OpenSSH:
Es el sustituto de herramientas antiguas como rlong, rcp, telnet, etc. El **programa cliente** es el paquete **ssh** y cuya **configuración** como cliente se encuentra en **/etc/ssh/ssh_config**.
Para conectar por ssh usamos: ssh usuario@ip
Ssh es una conexión segura y datos transmitidos están protegidos con una fuerte encriptación.
Las claves se generan automáticamente por el servidor ssh y dependiendo de la encriptación se almacenan en distintos sitios:

Formato	Clave Privada	Clave Pública
RSA	/etc/ssh/ssh_host_rsa_key	/etc/ssh/ssh_host_rsa_key.pub
DSA	/etc/ssh/ssh_host_dsa_key	/etc/ssh/ssh_host_dsa_key.pub

Cuando un cliente conecta por primera vez, le pregunta si acepta la clave pública y si la acepta la almacenará en **/.ssh/know_hosts**, que puede incluirse en **/etc/know_hosts** para que valga **para todos los usuarios**.
Además de las claves del anteriores, cada usuario puede tener su clave pública y privada para garantizar su autenticidad y poder acceder sin necesitar de suministrar la contraseña de login.
Para acceder sin contraseña de login es necesario crear la clave pública y primaria, la pública debe incluirse en el archivo authorized_keys de la computadora de destino. Estas claves se generan mediante el comando ssh-keygen:

- **ssh-keygen -t dsa -b 1024**: crea una clave dsa de 1024 bits.
- **ssh-keygen -r rsa -b 4096**: crea una clave rsa de 4096 bits.

Formato	Clave privada	Clave pública
RSA	~/.ssh/id_rsa	~/.ssh/id_rsa.pub
DSA	~/.ssh/id_dsa	~/.ssh/id_dsa.pub

Estas claves **se almacenan en ~/.ssh/**, si suponemos que el usuario es lucioano con clave DSA la podemos meter en la otra máquina mediante:
cat ~/.ssh/id_dsa.pub | ssh luciano@192.168.1.1 "cat >> ~/.ssh/authorized_keys"
Por cuestiones de seguiridad es importante **ponerle** como **permiso 600.**
También podemos **evitar escribir la clave mediante el comando ssh-agent** que es como un anillos de clave que la almacenará, este quedará activo y **mediante ssh-add podemos incluir la clave de usuario.**

Túneles encriptados:
Ssh además de abrir sesiones remotas de shell puede usarse como **vehículo para otras conexiones**, esto se conoce como túnel ssh. Una vez creado otro programa podrá comunicarse con la máquina remota por medio de este túnel de forma segura. Como por ejemplo con VNC que envía datos de forma insegura, pero podemos crear un túnel al puerto 5900 (VNC) de una máquina a otra para que de esta manera los datos estén protegidos.
Para **crear túneles: ssh -L puerto local:puerto remoto** ejemplo:
ssh -fNL 5900:localhost:5900 luciano@192.168.1.1 Con **-f** indicamos que debe ejecutarse en **segundo plano** y con **-N** indicamos que **no debe de abrirse sesión de shell** en la máquina remota.

X remoto por SSH:
Podemos abrir **ventanas** de una aplicación **remota mediante ssh usando -X**. Ej para mostrar vitualbox de una máquina remota:
ssh -X luciano@192.168.1.1 y luego ejecutar VirtualBox o podemos ponerlo todo en uno: **ssh -X luciano@192.168.1.1 "Virtualbox"**
Firma GnuPG:
Es una implementación de código abierto del estandar OpenPGP, con él es posible firmar y codificar archivos y mensajes paa garantizar su autenticidad. Esto se realiza mediante el concepto de clave pública y clave secreta (sólo la tiene el propietario).
El comando para realizar funciones de GnuPG es gpg:

- **gpg --gen-key:** genera clave, se recomienda DSA, también nos pide caducidad y longitud.
- **gpg –list-keys:** lista las claves que se encuentran en ~/.gnupg/.
- **gpg --export:** exporta la clave pública para dársela a los demás y puedan autenticar. Ejemplo: gpg --output archivodondeseguardara.gpg --export laIDquesequiereexportar. Esta clave es en formato binario, para sacarla como texto usamos –armor.
- **gpg --send-keys IDkey:** envia a un servidor de claves, por defecto es key.gnupg.net pero podemos cambiarlo en ~/gnupg/gpg.config.
- **gpg --import nombre.gpg**: importa la clave.
- **gpg --recv-keys idclave**: importa la clave desde el servidor de claves, esta clave se almacenará en ~/.gnupg/pubring.gpg
- **gpg --sing-key id_clave:** firma la clave para garantizar su autenticidad.

Firmar con gpg: gpg --output documento_que_sea.gpg --sign documento_que_sea. El archivo firmado se le enviará a otra persona que mediante la clave pública del autor podrá **asegurarse su autoría con**: gpg –output documento_que_sea –decrypt documento_que_sea.gpg.
Encriptar un archivo: gpg –output documento_que_sea.gpg –encrypt –recipient phess documento_que_sea. Para **desencriptarlo**: gpg –output documento_que_sea –decrypt documento_que_sea.gpg