**HOJA DE COMANDOS**

**1.Definiciones**

**Programa:** conjunto de datos e instrucciones que implementan un algoritmo.

**Proceso:** programa que está en ejecución y es independiente de otros procesos.

**Comando:** cadena de texto que identifica a un programa u orden.

* Suelen llevar argumentos
* - se denominan modificadores o flags

**Shell:** programa que deja ejecutar comandos.

* Lee una línea de comandos.
* Tokeniza
* Sustituye algunas cosas de la línea.
* Abre ficheros de redirecciones
* Crea los procesos para ejecutar los comandos descritos por la línea.

**Prompt:** texto que indica que el Shell está esperando una orden.

**Ruta absoluta:** serie de directorios desde el directorio raíz separados por barras. /home/alumnos/pepe/fichero.txt

**Ruta relativa:** serie de directorios desde el directorio actual. alumnos/pepe/fichero.txt

**Partición:** es como tener varios discos, cada partición tiene una etiqueta.

* Comandos: fdisk y parted/gparted permiten manipular las particiones de un disco.
* Los discos se direccionan mediante trozos denominados bloques.
* Existen dos tipos de esquemas:
  + **BIOS** (antiguo): esquema MBR: master boot record (512 bytes).
    - Cargador primario: 440 bytes
    - Tabla de particiones primarias: 4 entradas de 16 bytes
    - Direcciones de 32 bits
    - Limitado a 4 particiones (primarias)
  + **UEFI** (moderno): esquema GPT: GUID partition table.
    - Usa direcciones y tamaños de 64 bits
    - Primer bloque LBA 0: legacy MBR
    - Segundo bloque LBA 1: GPT header con punteros
    - El cargador está en una partición de tipo ESP (que es en realidad una partición FAT) que suele estar en /boot/efi

**Paquetes:** son un conjunto de ficheros .dev y son de tipo ar.

**Repositorio:** sitio donde hay paquetes con sus distribuciones.

* Hay una lista de repositorios en /etc/apt/sources.list y en los ficheros del directorio /etc/apt/sources.list.d (directorio).
* Dependiendo de cómo empiezan tipos:
  + Deb: binarios
  + deb-src: fuentes
  + remotos (https...)
  + locales (cdrom...)

**Runlevel:** es un concepto de System V, describe una configuración del sistema para un modo de operación. (T3 final)

**Pseudo-usuarios:** los que no pueden hacer login.

**/proc:** ficheros especiales sintéticos, no son ficheros de verdad.

**2.Directorios en GNU/Linux**

/bin tiene ejecutables.

/dev tiene dispositivos.

/etc tiene ficheros de configuración.

/home tiene los datos personales de los usuarios.

/lib tiene las bibliotecas (código) que usan los programas ejecutables.

/proc y /sys ofrecen una interfaz para interaccionar con el núcleo del sistema.

/sbin tiene los ejecutables del sistema.

/tmp sirve para almacenar los ficheros temporales, se borra en cada reinicio.

/usr existe por razones históricas (tamaño de almacenamiento) y contiene gran parte del sistema: contiene directorios similares a los anteriores (/usr/bin o /usr/lib), con los datos y recursos para los programas de usuario (no del sistema).

/var tiene los datos que se generan en tiempo de ejecución (cache, logs y otros ficheros que generan los programas).

/boot contiene los ficheros de arranque del sistema.

/media y /mnt puntos de montaje

/opt contiene ficheros para programas de terceros.

**3.Necesidades**

Man comando para buscar en el manual (q para salir) / comando –help versión corta del manual

. directorio actual

.. directorio padre

El tabulador completa nombres de ficheros.

‘\n’ salto de línea

‘\t’ un tabulador

Hacer root sudo –i / para salir exit

Variables: (locales y de entorno)

* Crear una variblesMivar=hola
* $mivar
  + Toquenizada: var=’hola que’ – echo $var – hola que
  + Sin toquenizar: echo “$var” – hola que
* Variables populares.
  + $PATH: la ruta de los programas.
  + $HOME: la ruta de tu directorio casa.
  + $USER: el nombre de usuario.
  + $PWD: la ruta actual del shell
  + $LANG: configuración de localización (locales).
  + $LC\_xxx: otras variables de localización (locales).

Formas de ejecutar un script:

* Bash miprograma.sh
* ./miprograma.sh (habiendo dado los permisos de ejecución)
* Añadirlo a una ruta: sudo cp/mv ruta
  + Crear ruta PATH=$PATH : ruta

Alias:

* Alias (sin argumentos): muestra los alias que hay definidos
* Alias NB=’comando’: crea un alias
* Unalias NB: borra un alias

Ejecución condicional:

* True && echo hola – hola
* False && echo hola –
* True || echo hola –
* False || echo hola – hola

**4.Scripts**

* #! Indicar que es ejecutable, en los nuestros: #!/bin/sh
* BUSCAR EN EL MANUAL: / busqueta enter n(saca la siguiente)
* Exit 0: salir con éxito, exit 1 salir con error. Mirarlo en la variable $?
* Redireccines:
  + < entrada estándar ej: grep H < fichero
  + > reedirecciona la salida estándar truncado el fichero
  + >> redirecciona la salida estándar poniéndolo al final del fichero
  + 2> redirecciona la salida error
* Entrada estándar 0, salida estándar 1, salida error 2.
  + Redireccionar salida estar a salida error 1>&2
* Entubamiento o pipelines (| = pipe): comando1 | comando2 – la salida de comando1 es la entrada de comando2
* Agrupaciones:
  + (comando; comando) en una subshell
  + {comando; comando} en la Shell actual
* Funciones: hola() {comandos}
* Parámetros posicionales:
  + $1, $2, $3…. Los parámetros que se pasan al script
  + $0 expande al nombre con el que se ha invocado al script
  + $# nº de parámetros
  + “$\*” expande a “$1 $2 ...”
  + $@ expande a los parámetros posicionales (igual que $\* pero separados)
  + ”$@” expande a ”$1” ”$2” ...
  + Shift: para a la siguiente si la hay.
    - Echo $1 shift echo $1
    - Script.sh Hola adiós
      * Hola
      * Adiós
* Sustitución de comando: $(comando) o `comando`
* $(( )) hace operaciones aritméticas
* $$ saca el pid de la sesión actual
* IF: depende del status de salida del comando, éxito—verdadero, fallo—falso

if (! negar) comando; then

comandos

elif comando; then

comandos

else

comandos

fi

* Case: patrones de globbing.

case palabra in

patrón1)

comandos

;;

patrón2)

comandos

;;

\*) en el resto de casos

Comandos

;;

Esac

* while:

while comando

do

comandos

done

* for

for i(variable) in palabra1/palabra2…. ó $(comando)

do

comandos

done

**5.Comandos**

Awk: awk ‘/selección(regex)/ {orden}’

* print: awk ‘{print $1}’
  + $0 línea entera
  + $1 primera columna, $2 segunda columna….
  + $NF última columna
  + NR – número de línea
  + Lenght(línea) – da la longitud de la línea
* printf(): permite control sobre el formato
  + awk ‘{printf(“%.1s\n”, $0)}’ 🡪 quedarse con el primer caracter
* se pueden declarar variables dentro: awk -v t=$total ‘{ print t}’
* -Fsepardor
* Awk ‘{system(“ls”)}’ – ejecutar un comando dentro de awk
* Awk ‘$1 ~ /[hH]/ {print $0}’: mira si la columna 1 contiene el regex
* Se pueden hacer comparciones => awk ‘($2 > 10.0 || $2 <0.0) {exit (1);}’
* Arrays asociativos
  + ps aux | awk ’{dups[$1]++} END{for (user in dups) {print user,dups[user]}}’
    - cuenta los procesos que está llevando a cabo cada usuario
      * lara 103
      * root 200
      * jane 2

Cal/ncal: muestran el calendario

Cat: escribe en su salida el contenido de un fichero

Cd: cambia el directorio actual

Clear: limpia el terminal

Cmp: Compara ficheros binarios byte a byte

Cp: copia ficheros

* Cp fichero directorio

Cut y paste: alternativas a awk y join.

Diff: compara ficheros línea a línea

Du –a . recorre un árbol de ficheros

Echo: escribe sus argumentos por su salida

* -n quita el salto de línea

Date: muestra la fecha

* Date +% (mirar opciones en el manual)

Exit: shell termina su ejecución

Export: convierte una variable en una variable de entorno – export mivar

File: da pistas sobre el contenido de un fichero

* File –mime-type fichero – archivo: tipo del fichero

Find . : recorre un árbol de ficheros

* -type f: coge solo los ficheros (aparte de f hay muchas más)
* -user username: coge solo los que sean del usuario indicado
* -Printf “%s %p” (size-nombre del fichero)

Grep: busca cadenas dentro de ficheros

* Egrep expresiones regulares
* Fgrep globbing
* -q no saca nada por su salida
* -v elimina la cadena de caracteres
* -i no hace distinción entre mayúsculas y minúsculas
* -c número de líneas en las que aparece la cadena

Gzip/gunzip: comprime/descomprime un fichero.

Head/tail: escribe las primeras/últimas líneas de un fichero

* -n nº líneas deseadas fichero

Join: relaciona dos documentos (tienen que estar ordenados) (tienen que tener algo en común en cuento a texto y el mismo nº de líneas)

Less: permite leer un fichero de texto en tipo scroll

Lsof: que procesos tienen que directorios/ficheros abiertos

Ls: lista el contenido de un directorio

* -l lista de forma larga
  + Directorio/fichero permisos dueño/grupo/resto
* -a incluye los que empiezan por .
* -i saca el nombre y el número de i-nodo

Mkdir: crea directorios

Mv: mueve ficheros/ cambia de nombre ficheros

* Mv fichero directorio
* Mv nombre\_viejo nombre\_nuevo

Od: escribe en su salida los datos de un fichero en distintos formatos

Pwd: ruta directorio actual

Printenv: muestra las varibles de entorno

Ps: procesos del sistema

* Ps aux: procesos en ejecución

Read: lee la entrada estándar

* Read VAR: lo almacena en la variable (VAR)
* Read -n 2 cadena – lee solo “ca”

Reset: restablece el estado del terminal

Rm: borra ficheros

Rmdir: borra directorios vacíos

Sed:

* -n sin el escribe todas las líneas después de procesarlas
* -E para regex
* $ hace referencia la última línea
* Sed ‘3q’ : salen del programa las 3 primeras líneas
* Sed –n –E ‘/regex/q’ imprime las lías que contengan ese regex
* Sed –n ‘1,2p’: imprime las líneas 1 y 2
* Sed –n -E ‘1,/regex/p’ : imprime desde la línea 2 hasta que aparce el regex
* Sed ‘1,2d’ : borra las líneas 1 y 2
* sed –E ‘s/regex/sustitución/’
  + **MUY IMPORTANTE** sed ‘s/(.).\*/\1.output/’ 🡪 el paréntesis guarda como variable
  + **Mayusculas a minúsculas**  sed ‘s/[A-Z]/L&/’
    - Tenemos hola, mar y dia
    - Resultado: h.output, m.output, d.output
    - Si fuese así sed ‘s/(a).\*/\1.output/’
    - Resultado: ha.output, ma.output, dia.output
  + normal solo sustituye la primera subcadena
  + sed –E ‘s/regex/sustitución/g’ (sustituye todas las subcadenas)
  + sed –E ‘s/[0-9]/X/g’ sustituye cada nº por una X
  + sed –E ‘s/([A-Za-z]+)[]+([0-9]\*)/NOMBRE:\1 NOTA:\2/g’
    - original: Lara 9
    - salida: NOMBRE: Lara NOTA: 9

seq: contar

Set: muestra todas las variables de shell y de entorno

Sort: ordena las líneas de un fichero

* -n primero letras luego números (normal al revés)
* -r ordena al revés
* -k\_nº columna que se tiene en cuenta para ordenar
* -t\_separador de columnas
* -u quita repetidos (como uniq)

Tar: crea un fichero con múltiples ficheros dentro (comprimidos o no)

Test: comprueba condiciones

* Test $a eq $b = [a$ -eq $b]
* -f si existe el fichero
* -d si existe el directorio
* -e si existe un fichero
* Strings:
  + -n String1: si la longitud del string no es 0
  + -z String1: si la longitud del string es 0
  + String1 = String2: si son iguales
  + String1 != String2: si no son iguales
  + String1: si la string no es nula
* Enteros:
  + Int1 –eq/ne In2: si son iguales
  + –gt/lt: mayor/menor que
  + –ge/le: mayor/menor o igual que

Top: muestra los procesos en ejecución refrescando sus datos

Touch: cambia la fecha de modificación de un fichero si existe, sino lo crea

Tr: traduce caracteres

* Mar(hola) – tr h H – mar(Hola)
* Tr –d “\n” elimina el salto de línea (-d = delete)

Uniq: elimina líneas contiguas repetidas

Unset: borra varibles

W: lista de los usuarios que están en el sistema / last: últimos usuarios q han entrado al sistema

Wc: cuenta los bytes, palabras y líneas de un fichero

* -c bytes
* -w palabras
* -l líneas
* -m caracteres

Who: muestra los usuarios que hay en el sistema

Whoami: muestra un nombre de usuario

Xargs:

$ echo a b c | xargs ls -l

-rw-rw-r-- 1 esoriano esoriano 2 mar 1 16:03 a

-rw-rw-r-- 1 esoriano esoriano 2 mar 1 16:03 b

-rw-rw-r-- 1 esoriano esoriano 2 mar 1 16:03 c

**Comandos sistemas de ficheros:**

* Mount: muestra los sistemas de ficheros que hay montados
* Df: espacios libres en los distintos sistemas de ficheros montados
* Netstat: ofrece información sobre las conexiones de red

**Comandos de permisos/grupos/usuarios:**

* Id: ids de usuario(uid) y grupo(guid)
* Passwd username: cambiar la contraseña de un usuario
* Adduser/deluser username: añadir/borrar un usuario.
* Adduser/deluser username groupname: añadir/eliminar a un usuario de un grupo.
* Addgroup/delgroup groupname: crear/borrar un grupo.
* Chmod: cambia los permisos de un fichero
  + –R de forma recursiva a todo el árbol
  + u(usuario)/g(grupo)/o(todos) +/- r/w/x fichero
  + Inicialmente el creador de un fichero es su dueño y grupo
  + Solo puede ejecutarlo root o el dueño
* Chown: cambia el dueño de un fichero.
* Chgrp: cambia el grupo de un fichero

**Comandos del sistema:**

* Uname –a
* Lsb\_release –a: informa que distribución se está realizando
* add-apt-repository nombre\_repositorio
* Dpkg: instala y borra ficheros de paquetes de directamente
  + dpkg -i xia 2.2-3 all.deb : instala xia localmente
  + dpkg -S fichero: ver a que paquete ya instalado pertenece el fichero
* apt update: actualiza la lista de paquetes disponibles
* apt upgrade: actualiza el sistema instalando/actualizando paquetes
* apt install xia. Instala xia y si ya lo tienes lo actualiza
* apt remove paquete
* apt list (regex/globing): lista todos los paquetes / los que contienen la cadena
* apt search regex: busca los paquetes que contienen dicha expresión regular en su nombre o descripción corta.
* apt show paquete: muestra detalles del paquete
* apt clean: libera espacio
* apt autoremove: borra los paquetes que no ha instalado el usuario y que ya no están en las dependencias de paquetes instalados.
* apt-file search fichero: buscar que paquete del repositorio contiene el fichero
* apt – search/get/cache: se baja paquetes de la red

**6.Globbing**

? un carácter cualquiera

\* cualquier secuencia de caracteres (hasta el carácter vacío)

[abc] cualquier carácter que está dentro de los corchetes

[^abc] ninguno de los caracteres que esté dentro de los corchetes

[a-e] [0-9] cualquier carácter que se encuentre entre esas dos

\ escapa un solo carácter especial

‘ ‘ escapan todo lo que está en su interior

“ “ escapan todo menos algunas sustituciones

**7.Regex**

. hace referencia cualquier carácter

\ escapa solo un caracter especial

[] conjunto de caracteres, ej: [a-zA-Z], [aeiou]

[^] no encaje con el conjunto

^ principio de línea

$final de línea

? encaja si aparece 0 ó 1 vez el regex que lo procede

\* encaja si aparece 0 ó más veces el regex que lo procede

+ encaja si aparece 1 o más veces el regex que lo procede

| combinar regex, o una o la otra ¡RECUERDA ESCAPAR \|!

() agrupa regex

* ^Hola\* encaja con Hola, Hol, Holaaa, Holae
* ^Hola\*$ encaja con Hola, Hol, Holaaa
* ^Hola+ encaja con Hola, Holaaa, Holae
* ^Hola+$ encaja con Hola, Holaaa
* ^Hol(aa)+ encaja con Holaaa

**8.Controles**

Control izquierdo

↑ repite los comandos ejecutados anteriormente en la shell.

Ctrl+r deja buscar comandos que ejecutamos hace tiempo.

Ctrl+c mata el programa que se está ejecutando.

Ctrl+z detiene el programa que se está ejecutando.

Ctrl+d termina la entrada (o manda lo pendiente).

Ctrl+a: mueve el cursor al principio de la línea.

Ctrl+e: mueve el cursor al final de la línea.

Ctrl+w: borra la palabra anterior en la línea.

Ctrl+u: borra desde el cursor hasta el principio de la línea.

Ctrl+k: borra desde el cursor hasta el final de la línea.

Ctrl+s: congela el terminal / Ctrl+q lo descongela.

Ctrl+l: limpia el terminal.

**9.Vi**

Tiene dos modos: modo inserción (para escribir) y modo comando.

* i pasa a modo inserción
* ESC pasa a modo comando

En modo comando:

* :q! sale del editor sin guardar
* :x salva el fichero y sale (también se puede con: wq)
* :w salva el fichero
* :número se mueve a esa línea del fichero
* i inserta antes del cursor
* a inserta después del cursor
* o inserta en una línea nueva
* dd corta una línea
* p pega la línea cortada anteriormente
* h, j, k, l mueve el cursor a izquierda, abajo, arriba, derecha

**10.Gestión de usuarios y permisos**

* Para el kernel, los usuarios y grupos de Unix son números (id, usuarios: uid, grupos gid).
* Etc/passwd: fichero de texto con la traducción entre nombres y uids.
* La contraseña se guarda en etc/shadow sin permisos de lectura para nadie.
* Etc/group contiene información sobre los grupos.
* Ejemplos de control de credenciales:
  + jose ALL = (root, bin : operator, system) ALL
    - jose puede, en cualquier máquina
    - adquirir el UID de root y bin
    - adquirir el GID de operator y system
    - para ejecutar cualquier comando
  + ramon mono = NOPASSWD: /bin/kill, PASSWD: /bin/ls, /usr/bin/lprm
    - ramon puede, en la máquina mono
    - adquirir el UID de root para ejecutar kill sin proporcionar contraseña
    - para ejecutar ls y lprm proporcionando contraseña

**11.Ficheros y comandos avanzados**

Metadatos: datos sobre un fichero, no del fichero (ej: nombre y permisos).

- Las entradas de un directorio son nombres de fichero y un número de los identifica(i-node). - - Distintas entradas pueden tener el distinto nombre, pero que referencien a un mismo i-node. - Directorio raíz => i-nodo 2

**Enlaces duros:** otro nombre para el fichero, es decir, otra entrada de directorio para ese i-node. El contador de i-node(segunda columna ls -l) se incrementa/decrementa al añadir/eliminar un nuevo nombre al fichero. El fichero solo se borrará si su contador es 0.

* El comando ln permite crear un enlace duro (ln a b)

**Enlaces simbólicos o blandos:** se trata de otro fichero especial (symlink), cuyos datos contienen la ruta al fichero enlazado. Se rompen si el fichero original se borra.

* Se crean con ln –s a b
* 2371883 lrwxrwxrwx 1 esoriano esoriano 1 mar 21 13:43 **b -> a**

**Pipes con nombre:** llamados fifo, se crean con mkfifo

**Dispositivos:** se representan como ficheros. Se crean con mknod.

* Comando dd: copiar datos entre dispositivos

Tipos de ficheros:

* Enlace simbólico => l
* Pipes con nombre => p
* Dispositivos => c o b
* Conexiones de red => s

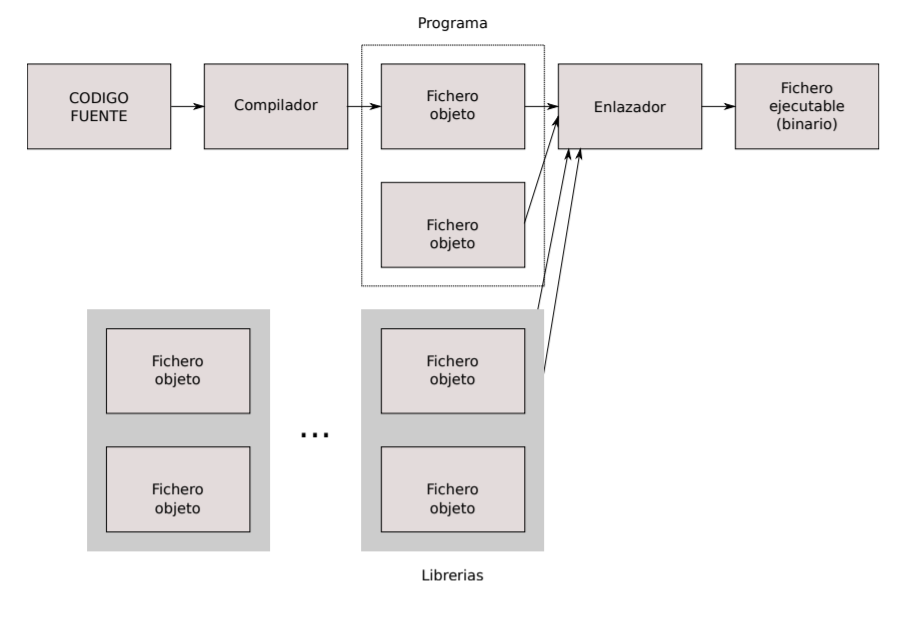
**/proc** => tiene un directorio por cada proceso en ejecución (el nombre del directorio es su PID: /proc/322)

* Cwd => enlace simbólico al directorio actual
* Exe => enlace simbólico al ejecutable
* Fd => ficheros que tiene abiertos

**Logs:**  ficheros de bitácora, están en /var/log. Cuando un fichero llega a un tamaño o edad se comprime y se renombra.

**12.Compilación, carga y depuración de programas**

Librerías: funciones ya programadas que utilizan los programas, pueden ir dentro del binario final (enlazado estático = tiempo de compilación .a) o en un fichero binario separado (enlazado dinámico = tiempo de ejecución .so) => formato Elf



El **compilador** resuelve todos los símbolos que hay en el fichero y para cada uno crea una tabla. Los símbolos ya resueltos para depurar y el resto se resuelven posteriormente con las librerías(enlazador) => genera los ficheros objeto, código relocalizarble y posiblemente con símbolos sin resolver.

Para cargar un programa hay que copiarlo en un lugar concreto de memoria, si tiene enlazado dinámico hay que cargar también las librerías dinámicas (ld) y resolver los símbolos.

Profiling: medir el uso de recursos, memoria…

**13.Arranque del sistema**

La secuencia es:

* Firmware: software que vine incrustado en el hardware.
  + UEFI
    - El cargador del sistema operativo es una aplicación EFI.
    - efibootmgr –v: muestra las variables de UEFI
  + Inicializar el hardware
  + Iniciar el arranque del sistema
  + Ofrece una interfaz para operar con algunos componentes del hardware
* Cargador primario
  + Objetivo: seleccionar y encontrar la imagen del kernel que se quiere arrancar en una partición del disco, cargarla en memoria y saltar a ella
  + Más común GRUB
    - Gestor de arranque: permite seleccionar y lanzar otros cargadores para arrancar distintos sistemas operativos.
    - Tiene su propio intérprete de comandos
* Cargadores secundarios (= cargador primario)
* Kernel
  + …………….. (diapositivas T3 todo lo que hace)
  + Initrd: sistema de ficheros que se usa para poder arrancar el sistema completo (initramfs). Todos estos ficheros están en /boot/initrd.img-\*
    - La raíz(/) se monta de forma temporal porque el kernel es modular y exsiten muchos drivers distintos para los tipos de hardware(algunos necesitan comandos).
* Área del usuario (init)
  + El proceso con PID 1 quedará ejecutando el programa /sbin/init hasta que se apague la máquina.
  + Todos los procesos de usuario son hijos de init
  + Objetivos:
    - Montar los sistemas de ficheros configurados.
    - Crear los procesos que terminan ejecutando la interfaz gráfica, programas de login, etc.
    - Iniciar y parar los servicios (demonios).
    - Liberar recursos cuando mueren procesos.
    - Esperar a que se ordene el apagado o reinicio del sistema
  + Systemd: colección de demonios, bibliotecas, programas y componentes del núcleo.
    - Comandos (hay que ejecutarlos como root):
      * systemctl: permite controlar y dar órdenes a systemd
      * systemd-analyze: comando que permite analizar y depurar systemd
    - Unit: cualquier cosa que puede manejar systemd, son ficheros de tipo texto plano.
      * Comando + nombre unit: \*(sin nada detrás)
        + systemctl start/stop/restart: arranca/para/reinicia el servicio.
        + systemctl enable/disenable: habilitar o deshabilitar un servicio (se arranca cuando se inicia el sistema).
        + systemctl status: nos muestra el estado del servicio.
        + \*systemctl list-units: muestra todas las units activas.
        + \* systemctl list-unit-files: muestra todos los ficheros de las units.

Static: no puede ser habilitada, solo realiza una acción, dependencia de otra unit

Masket: no puede ser arrancada

* + - * + systemctl show: muestra las propiedades de la units.
        + systemctl list-dependencies: las dependencias de la unit.
        + systemctl edit –full: editar el fichero de la unit
        + \*systemctl daemon-reload: recarga systemd.
        + systemctl halt: apaga el sistema.
      * /lib/systemd/system // pkg-config systemd --variable=systemdsystemunitdir: instalar units // no modificables.
      * /etc/systemd/system --variable=systemduserunitdir// si modificables
      * Las units dependen unas de otras, tipos de dependencia:
        + Requires: es estricta, se activará y si falla la activación de la dependencia, no se puede activar esta unit.
        + Wants: se intentará activar la dependencia, pero si falla no importa.
        + Requisite: debe estar activa ya, si no lo está, falla la activación de esta unit.
        + Conflicts: si está activa, se desactivará, no es compatible con esta unit.
      * Tipos
        + Service units: controlan los servicios (demonios).
        + Mount units: controlan los sistemas de ficheros.
        + Target units: controlan otras units.
        + Path units: vigilan rutas para reaccionar si cambian ficheros/directorios.