**1.1. TEMA: Administración Linux**

**1.1.1. Intérprete de comandos (Shell):**

Shell:

* Interfaz del sistema entre usuario/aplicaciones y las llamadas al sistema operativo.
* Un programa del sistema operativo con los mismos privilegios que cualquier otro.

Símbolos:

* $ → Usuario normal.
* # → Administrador/root.

Tipos de acceso:

* Local: Terminales de texto y una terminal gráfica.
* Remoto: A través de la red: telnet, rlogin, ssh…

Múltiples Shell en los sistemas Unix/Linux:

* bash: Bourne Again Shell.
* csh: C Shell.
* ksh: KornShell.
* tcsh: Tenex Csh, tiene mejoras sobre csh.
* zsh: Extensión de bash con características de ksh y tcsh, por defecto desde macOs 10.15.
* Es importante conocer cuál estamos usando.

Comando man:

* Información sobre cualquier comando.
  + man <comando>
* Los manuales se encuentran en /usr/share/man.
* Configuración en /etc/manpath.config.

**1.1.2. Sistema de ficheros:**

Comando ls:

* Listar archivos.
* Parámetros:
  + l → Mostrar información adicional (fecha, tamaño, permisos, …).
  + a → Mostrar todos los archivos, incluidos los ocultos.
  + r → Lista en orden inverso.
  + t → Ordenar cronológicamente ascendente.
  + h → Mostrar tamaños de ficheros en KB o MB en vez de bytes.

Comandos para navegar por el sistema de ficheros:

* pwd → Mostrar directorio actual.
* cd → Cambiar directorio.
* mkdir → Crear directorio.

Manipulación de archivos:

* cp → Copiar archivos.
* mv → Mover archivos (o renombrar). mv <ruta-fichero-origen> <destino>
* rm → Borrar archivos (o directorios con el parámetro -r).
* ln → Crear enlaces a archivos (-s para simbólico). ln -s <ruta-fichero-origen> <ruta-enlace>
* whereis → Ubicación de un comando
* find → Buscar archivos. find <carpeta-base> -name <nombre-archivo>

Contenido de archivos

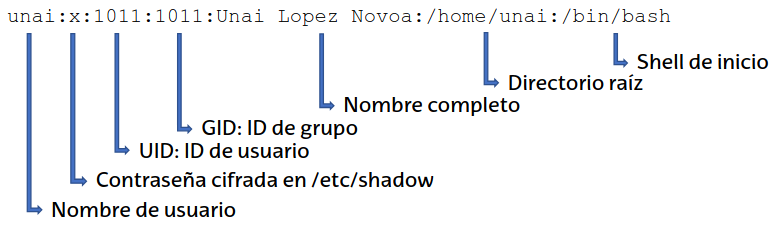
* cat/more/less → Mostrar contenidos de un archivo
* wc → Contar palabras (o líneas con el parámetro -l)
* head/tail → Mostrar N primeras/últimas líneas (utilizando -n)
* grep → Buscar patrón de texto en un archivo. grep casa miArchivo.txt
* cut → Muestra secciones concretas de un archivo. cut -d “ ” -f2 a.txt
  + Lee a.txt, separado por “ ”, y devuelve la 2ª columna.
* tar → Comprimir/descomprimir archivos/carpetas.
  + Comprimir: tar cfvz carpeta.tgz miCarpeta
  + Descomprimir: tar xfvz carpeta.tgz
* sort → Ordena las líneas de un archivo alfabéticamente.

**1.1.3. Usuarios:**

En Unix, los usuarios están organizados en grupos.

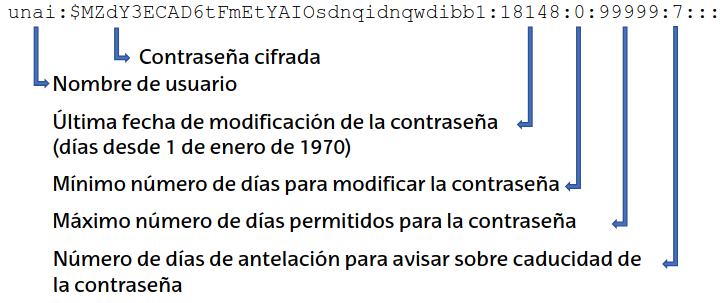
Los ficheros de /etc/passwd y /etc/group tienen la información sobre los usuarios y grupos.

* Cada línea tiene diferentes campos separados por el carácter “:”.



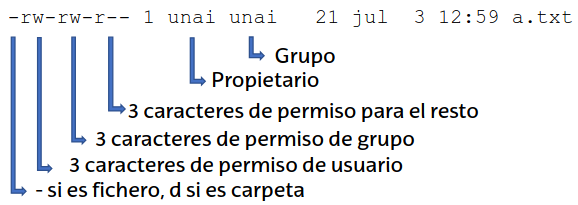
/etc/shadow:

* Tiene la contraseña del sistema.
* Hacen falta permisos root para leerlo.
* Cada línea tiene esta forma:



Permisos de los usuarios:

* Usuario:
  + Solamente permisos de escritura en su raíz (/home/<usuario>).
  + Permisos sobre algunos directorios temporales como /tmp.
* Super-usuario (sysadmin):
  + Acceso a todo el sistema de archivos.
* Ejemplo de permisos:



Comandos básicos de gestión de usuarios:

* whoami → Muestra el nombre del usuario actual.
* who → Muestra los usuarios conectados al sistema.
* w → Equivalente a who, con algo más de información.
* passwd → Cambiar contraseña del usuario actual.
* write → Escribir un mensaje a otro usuario.
* useradd → Crear usuario en el sistema.
* adduser → Script “asistente” para crear un usuario.

Gestión de permisos

* chmod → Modificar permisos de un fichero o carpeta.
* chown/chgrp → Modificar UID/GID de un fichero.

**1. ariketa:**

* chmod u-r Ej1.txt
* wc ./.bashrc
* cut -d: -f1 /etc/passwd
* grep "/bin/bash" /etc/passwd
* ln -s /bin/bash /tmp/miBash

**1.1.4. Variables de entorno:**

Función:

* Para guardar valores en una sesión de Shell.
* Para leer el contenido: echo $VARIABLE. (Ej.: echo $HOME).

2 tipos:

* De usuario:
  + Se mantienen durante la sesión activa, después se eliminan.
  + Se listan con el comando env.
  + Ej.: $HOME, $LD\_LIBRARY\_PATH
* Del sistema:
  + Están en todas las sesiones del sistema.
  + Se listan con el comando set.
  + Ej.: $BASH, $HOSTNAME

Se crean diferente según la Shell:

* Shell tipo sh:
  + Ejemplos: Bash, ksh.
  + Comando export.
    - Ej.: export MIVARIABLE=4
* Shell tipo csh:
  + Ejemplos: tcsh, zsh.
  + Comando setenv.
    - Ej.: setenv MIVARIABLE 4

Algunas variables de entorno comunes:

* $PATH → Listado de directorios donde están los binarios de los comandos. Al ejecutar un comando (Ej.: ls), se busca aquí.
* $HOME → Directorio raíz del usuario actual.
* $TERM → Tipo de terminal utilizado para conectar al sistema.
* $SHELL → Tipo de Shell de la sesión (Ej.: Bash).
* $PWD → Directorio actual.

Hay ficheros del sistema que permiten configurar la Shell.

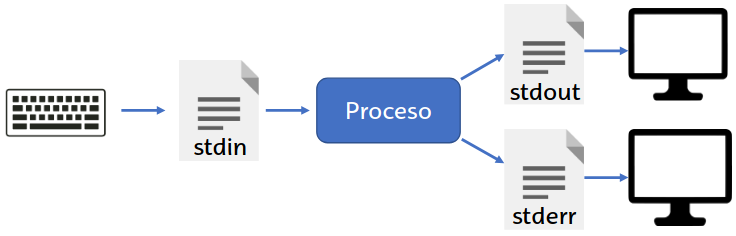
Podemos definir variables de entorno en ellos para que estén siempre disponibles en nuestras sesiones.

* En Bash, en orden de relevancia:
  + /etc/bashrc → /etc/profile→$HOME/.bashrc→$HOME/.bash\_profile
* En zsh:
  + /etc/zshrc → /etc/profile→$HOME/.zshrc→$HOME/.zprofile

**1.1.5. Redirección y Pipes:**

En Unix, cada proceso tiene asociado por defecto 3 ficheros para gestionar entrada y salida:

* stdin → Teclado (entrada estándar).
* stdout → Pantalla (salida estándar).
* stderr → Pantalla (salida de errores).



Redirección:

* Utilizar un fichero en lugar del teclado/pantalla para la entrada/salida.
* Redirección de stdin:
  + Utilizar un fichero en lugar del teclado para la entrada de datos.
  + Ej.: sort < a.txt
* Redirección de stdout:
  + cat a.txt > b.txt → Abre/Borra b.txt y escribe la salida de cat.
  + cat a.txt >> b.txt → Añade la salida de cat al final de b.txt.
* Redirección de stderr:
  + cat a.txt 2> b.txt → Escribe en b.txt los errores que genera cat.
* Redirección de stdout y stderr:
  + cat a.txt &> b.txt → Escribe en b.txt la salida de cat y sus errores.

Un Pipe (“tubería”) permite redirigir la salida de un comando a la entrada de otro:

* Ejemplo: cat a.txt | grep casa | wc -l
* Contaría el número de líneas de a.txt que contienen “casa”.

En Bash se puede concatenar la ejecución de varios comandos seguidos:

* Ejemplo: ls -l; cd ..; ls -l
* Alternativamente: ls -l && cd .. && ls -l

**1.1.6. Shell scripting:**

Guión de Shell/Shell script:

* Un conjunto de comandos que se ejecutan con un objetivo concreto.
* Su forma más simple: un fichero de texto con un comando por línea.



Para ejecutar un script:

* Si el script no tiene permisos → bash miScript
* Modificar los permisos y ejecutarlo como un binario → chmod +x miScript; ./miScript

Entrada/salida en un script:

* read → Para leer el teclado.
* echo/printf → Para escribir en la pantalla (stdout).

Un script puede recibir parámetros y convertirlos en variables:

* $1 … 9 → Los parámetros, el número indica su posición
* $0 → Contiene el nombre del script
* $# → Número de parámetros recibidos
* $? → El PID del proceso que está ejecutando el script
* $\* → Contiene todos los parámetros ($1 + $2 + …)

Control de flujo:

* Sentencia “if - else”:
  + En /bin/bash: if [[ <condición> && <condición> ]]; then
  + En /bin/sh: if [ <condición> ] && [ <condición> ]; then



* Operadores de comparación:

| **Strings** | **Numéricos** | **Verdadero si** |  | **Operador** | **Verdadero si** |
| --- | --- | --- | --- | --- | --- |
| x = y | x -eq y | x es igual a y |  | -d carpeta | Carpeta existe |
| x != y | x -ne y | x es diferente a y |  | e fichero | Fichero existe |
| x < y | x -lt y \* | x es menor que y |  | -s fichero | Fichero existe y no está vacío |
|  | \*: -gt, -le, -ge | equivalen: > , <= , >= |  | -r fichero  -w fichero | Usuario tiene permiso de  lectura / escritura en el fichero |

* Sentencia case:



Bucles: Se puede utilizar el bucle for para iterar sobre (entre otros):

* Archivos:



* Números o elementos concretos:



* Secuencias de números:



Variables:

* Todas las variables en Bash se consideran Strings.
* Se puede hacer aritmética con variables Bash:
  + Utilizar (( )).
* Se pueden crear listas de variables:
  + Utilizar list.

Expresiones regulares:

* Se usan para buscar texto que corresponda a un patrón.
* Un patrón se construye con literales y caracteres especiales.
* Ejemplos de patrones:
  + “casa” → Encontraría “casa”.
  + “c([a-z])sa” → Encontraría “casa”, “cbsa”, “ccsa”…
  + “c([a-z]+)sa” → Encontraría “casa”, “caasa”, ”cxyzsa”…
  + “c(\d\*)sa” → Encontraría “csa”, “c101sa”…

| **Símbolo** | **Significado** |
| --- | --- |
| . | Cualquier carácter. |
| [] | Caracteres definidos entre []. |
| \d | Cualquier dígito. |
| \* | Una, ninguna o más del elemento precedido. |
| + | Uno o más del elemento precedido. |

Funciones:

* Se pueden crear funciones para ordenar el código.
* Se le pueden pasar parámetros, que se recogen en la función como $1, $2, … en base al orden.
* Para devolver un valor de la función, utilizar echo (return se usa para finalizar el programa).

