

ADMINISTRACIÓN LINUX

INTÉRPRETE DE COMANDOS (SHELL)

- Interfaz entre el usuario/aplicaciones y las llamadas al sistema operativo (SO)
- Programa del SO con los mismos privilegios
- Símbolo de uso:
 - Usuario normal → \$
 - Root → #
- Tipos de acceso
 - Local → Terminales de texto (Ctrl+Alt+F1...6) y terminal gráfica (Ctrl+Alt+F7)
 - Remoto → A través de red: telnet, ssh, rlogin
- Múltiples Shell (importante saber cuál usamos):
 - bash → Bourne Again Shell
 - csh → C Shell
 - ksh → KornShell
 - tcsh → Tenex Csh (mejoras sobre csh)
 - zsh → Extensión de bash con características de ksh y tcsh zsh (shell por defecto desde macOS 10.15)

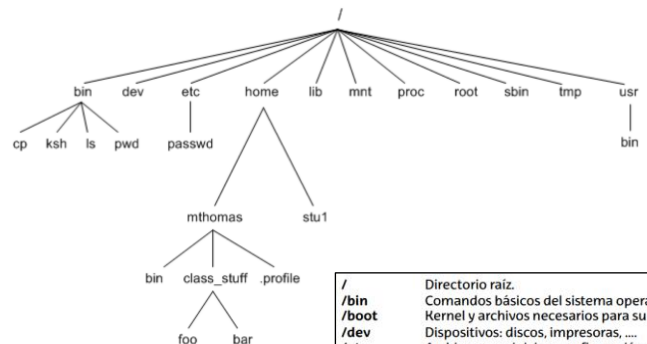
```
ncampillay@Pc93:~$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/dash
/bin/bash
/bin/rbash
ncampillay@Pc93:~$
```

COMANDOS

- man → Muestra información sobre un comando, función...
- Sintaxis: man <comando>
- Los manuales se encuentran en /usr/share/man
 - Se busca alfabéticamente y se muestra la primera entrada que coincida
- Configuración en /etc/manpath.config

SISTEMA DE FICHEROS

- Estructura de árbol
- Tipo de path:
 - Absoluta: /home/user/a.txt
 - Relativa: ./a.txt
- Los archivos ocultos empiezan por "."
 - Para verlos: ls -a
- Unidades externas (CD, HDD externo...)
 - Se pueden asociar a cualquier posición jerárquica
 - El mismo programa puede tratar archivos internos y dispositivos externos de forma diferente



/	Directorio raíz.
/bin	Comandos básicos del sistema operativo.
/boot	Kernel y archivos necesarios para su carga.
/dev	Dispositivos: discos, impresoras, ...
/etc	Archivos para inicio y configuración del sistema.
/mnt	Puntos de montaje temporales.
/lib	Librerías compartidas.
/home	Directorios raíz por defecto de los usuarios.
/opt	Paquetes de software opcional (no siempre)
/root	Directorio raíz para el superusuario.
/sbin	Comandos críticos del sistema operativo.
/proc	Información de procesos en ejecución.
/tmp	Archivos temporales.
/usr	Recursos y software de los usuarios.
/usr/local	Software instalado por los usuarios.
/var	Datos específicos y configuración del sistema.

COMANDOS

- Listar archivos: ls
 - l → Información adicional (fecha, tamaño, permisos...)
 - a → Todos los archivos
 - r → Orden inverso
 - t → Listar en orden cronológico ascendente
 - h → Mostrar tamaños de ficheros en KB o MB (en vez de bytes)
- Navegar por el sistema de ficheros:
 - pwd → Directorio actual
 - cd → Cambiar directorio
 - mkdir → Crear directorio
- Manipulación de archivos
 - cp → Copiar archivos
 - mv → Mover archivos (o renombrar)
 - rm → Borrar archivos (o directorios con -r)
 - ln → Crear enlaces a archivos (parámetro -s para simbólico)
 - find → Buscar archivos
- Contenido de archivos
 - cat/more/less → Mostrar contenidos de un archivo
 - wc → Contar palabras (o líneas con -l)
 - head/tail → Mostrar N primeras/últimas líneas (-n)
 - grep → Buscar patrón de texto en un archivo
 - cut → Muestra secciones concretas de un archivo (cut -d " " -f2 a.txt)
 - tar → Comprimir/descomprimir archivos/carpetas
 - Comprimir: tar cfvz carpeta.tgz miCarpeta
 - Descomprimir: tar xfvz carpeta.tgz
 - sort → Ordena las líneas de un archivo alfabéticamente

ATAJOS

- Para autocompletar Shell → Tabulador (si hay varias rutas se mostrarán todas)
- Expresiones regulares:
 - * → Reemplazar todos los caracteres
 - ? → Reemplazar un único carácter
 - [] → Reemplazar por un rango numérico
 - ~ → Atajo para el directorio raíz
- Si hiciera falta buscar un carácter de estos en un archivo, escribirlo precedido por \ o rodeado por ""

HISTORIA DE COMANDOS

- Histórico de comandos → history
 - Se puede volver a ejecutar un comando escribiendo !<nº comando>
 - !! ejecuta de nuevo el último comando
 - Con los cursores arriba/abajo se navega lo más reciente
 - Pulsar Ctrl+r y empezar a escribir para que la Shell autocomplete comandos históricos

USUARIOS

- Los usuarios están organizados en grupos
- Los ficheros `/etc/passwd` y `/etc/group` contienen la información sobre los usuarios y grupos
- Cada línea tiene diferentes campos separados por el carácter:

```
unai:x:1011:1011:Unai Lopez Novoa:/home/unai:/bin/bash
```

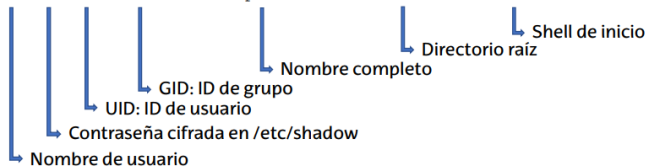


Diagram illustrating the fields of a line in `/etc/passwd`:

- Nombre de usuario
- Contraseña cifrada en `/etc/shadow`
- UID: ID de usuario
- GID: ID de grupo
- Nombre completo
- Directorio raíz
- Shell de inicio

- `/etc/shadow` contiene contraseñas del sistema (root para leerlo):

```
unai:$MzdY3ECAD6tFmEtYAI0sdnqidnqwdibbl:18148:0:99999:7:::
```

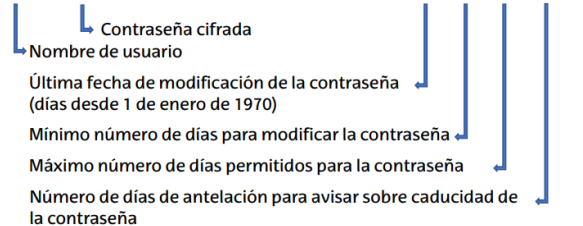


Diagram illustrating the fields of a line in `/etc/shadow`:

- Nombre de usuario
- Contraseña cifrada
- Última fecha de modificación de la contraseña (días desde 1 de enero de 1970)
- Mínimo número de días para modificar la contraseña
- Máximo número de días permitidos para la contraseña
- Número de días de antelación para avisar sobre caducidad de la contraseña

- La seguridad del sistema de ficheros se gestiona con usuarios, grupos y permisos para cada uno
- Cada fichero (y carpeta) tiene un único propietario y unos permisos de acceso
- Permisos:
 - `r` → Lectura
 - `w` → Escritura
 - `x` → Ejecución
- Cada permiso se configura para:
 - Usuario (`u`) → El propietario del fichero
 - Grupo (`g`) → Usuarios del grupo al que pertenezca el propietario
 - Resto (`o`) → Resto de usuarios del sistema
- Generalmente, los usuarios sólo tienen permisos de `W` en su directorio raíz + algunos directorios temporales como `/tmp`
- El super-usuario (o `sysadmin`) tiene acceso todo el sistema de archivos

```
-rw-rw-r-- 1 unai unai 21 jul 3 12:59 a.txt
```

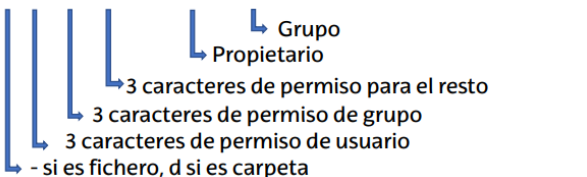


Diagram illustrating the fields of a file listing line:

- si es fichero, d si es carpeta
- 3 caracteres de permiso de usuario
- 3 caracteres de permiso de grupo
- 3 caracteres de permiso para el resto
- Propietario
- Grupo

COMANDOS

- Básicos de gestión de usuarios
 - `whoami` → Usuario actual
 - `who/w` → Usuarios conectados al sistema
 - `passwd` → Cambiar contraseña del usuario actual
 - `write` → Escribir un mensaje 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

VARIABLES DE ENTORNO

- Guardar valores en una sesión de Shell
- Para leer el contenido: echo \$VARIABLE
 - echo \$HOME
- 2 tipos:
 - Del usuario → Se mantienen durante la sesión activa, después se eliminan
 - Se listan con el comando env
 - P.e. \$HOME, \$LD_LIBRARY_PATH
 - Del sistema → Están en todas las sesiones del sistema
 - Se listan con el comando set
 - P.e., \$BASH, \$HOSTNAME
- Depende de la Shell, se crean de manera diferente
 - Shell tipo sh
 - P.e. Bash, ksh
 - Export → export MIVARIABLE=4
 - Shell tipo csh
 - P.e. tcsh, zsh
 - Setenv → setenv MIVARIABLE 4
 - Desaparecen al finalizar la sesión

VARIABLES DE ENTORNO COMUNES

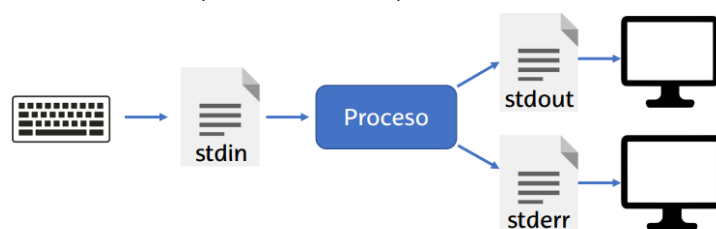
- \$PATH → Listado de directorios donde están los binarios de los comandos
 - Al ejecutar un comando (p.e. ls), se busca aquí
- \$HOME → Directorio raíz del usuario actual
- \$TERM → Tipo de terminal utilizado para conectar al sistema
- \$SHELL → Tipo de Shell de la sesión, p.e. Bash.
- \$PWD → Directorio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

DEFINICIÓN DE VARIABLES

- Hay unos ficheros en el sistema que permiten configurar la Shell (escribiéndolas aquí estarán disponibles siempre en nuestras sesiones)
- En Bash hay varios, en orden de relevancia:
 - /etc/bashrc → /etc/profile → \$HOME/.bashrc → \$HOME/.bash_profile
- Igualmente, en zsh:
 - /etc/zshrc → /etc/profile → \$HOME/.zshrc → \$HOME/.zprofile

REDIRECCIÓN Y PIPES

- Cada proceso tiene asociado 3 ficheros que sirven 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 → Ejemplo: `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 de cat
 - Redirección de stdout y stderr:
 - `cat a.txt &> b.txt` → Escribe en b.txt la salida y errores de cat
- Un Pipe permite redirigir la salida de un comando a la entrada de otro
 - Ejemplo: `cat a.txt | grep casa | wc -l`
- Además, en Bash se puede concatenar la ejecución de varios comandos seguidos
 - Ejemplo: `ls -l; cd ..; ls -l`
 - Alternativamente: `ls -l && cd .. && ls -l`

BORRAR+CREAR	AÑADIR	CONCATENAR
>	>>	

SHELL SCRIPTING

- Conjunto de comandos que se ejecutan con un objetivo concreto
- Su forma más simple: un fichero de texto con un comando por línea
- Si el script no tiene permisos: `bash miScript`
- Alternativamente, modificar los permisos: `chmod +x miScript;`
`./miScript`
- Entrada/salida en un script
 - `read` → Leer del teclado
 - `echo/print` → Escribir por pantalla (stdout)
- Puede recibir parámetros → Se convierten en variables: `$1 .. 9`
 - 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 + ...`)
- Sentencia `if – else`:
 - en `/bin/bash` `if [<condición> && <condición>]; then`
 - en `/bin/sh` `if [<condición>] && [<condición>]; then`
- Sentencias:

```
#!/bin/bash
echo "Hola"
echo "Mundo"
```

La 1ª línea se llama Shebang, indica la ruta del Shell a usar

```
#!/bin/bash
if [ `whoami` = "unai" ]; then
    echo "El usuario actual es unai"
else
    echo "El usuario actual no es unai"
fi
```

Comillas `` para ejecutar un comando

Atención al "fi" para cerrar el bloque "if"

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

```
usu=$1
case $usu in
    "unai") echo "El usuario es Unai";;
    "mikel") echo "El usuario es Mikel";;
    "jon") echo "El usuario es Jon";;
esac
```

- Bucles:

- Archivos:

```
for archivo in `ls`; do
    echo $archivo;
done
```

- Número o elementos concretos:

```
for i in 1 3 5 9 11; do
    echo $i;
done
```

- Secuencias de números:

```
for i in {1..8};do
    echo $i;
done
```

- Variables → Se consideran Strings

- Se puede hacer aritmética
 - Utilizar (())
 - Se pueden crear listas de variables
 - Utilizar list

- Expresiones regulares → Buscar texto que corresponda a un patrón

- Literales + caracteres especiales. Ejemplos de patrones:
 - “casa” → “casa”
 - “c([a-z])sa” → “casa”, “cbasa”, “ccsa”,...

```
a=1
b=3
c=$a+$b
d=$(( $a+$b ))
echo $c $d
```

Concatenación de strings

Suma aritmética

```
aa=11
bb=22
cc=33
list=(aa bb cc)
```

Muestra todos los elementos

```
echo ${list[0]}
echo ${list[0]}
```

Muestra el 1er elemento

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

- Funciones

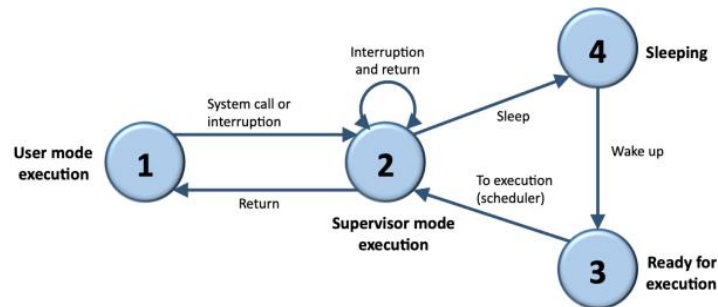
- Se pueden crear funciones para ordenar el código
 - Se le pueden pasar parámetros → \$1, \$2, ... en orden
 - Para devolver un valor de la función, utilizar echo y no return
 - return → finalizar el programa

```
function suma()
{
    OP1=$1
    OP2=$2
    echo $(( $OP1+$OP2 ))
}

A=33
B=44
res=$(suma $A $B)
echo "La suma de $A y $B es $res"
```

GESTIÓN DE PROCESOS

- Proceso: secuencia de instrucciones y datos almacenados en memoria
 - Se identifican con un número único: PID
 - Estados de un proceso:



- Se organizan en árbol → El proceso raíz es init
 - Cada proceso (excepto init) tiene un padre
 - El kernel tiene control de todos los procesos del sistema
- Para cada proceso, su PID se identifica con el UID del usuario al que le pertenece
- Podemos lanzar procesos en primer y segundo plano
 - Primer plano: la terminal se bloquea hasta que el proceso termina
 - Segundo plano: la terminal devuelve el control inmediatamente
- Para lanzar un proceso en 2º plano, añadir & al final
 - Ejemplo: ./miScriptLargo &
- El comando nohup (No Hang Up) mantiene un proceso vivo aunque se cierre la sesión
- Al combinar nohup y &, podemos dejar un proceso en ejecución y cerrar nuestra Shell
 - Ejemplo: nohup ./miScriptLargo &
- En la carpeta /proc se encuentra la información de los procesos (usada por el kernel)
 - Cada proceso tiene una carpeta:

```
unai@unai-server:/proc$ ls
1 13 1742 205 2365 252 310 43 821 953 99 ipmi sched_debug
10 1300 18 20537 2372 2530 32 44 822 957 acpi irq schedstat
```

- Cada carpeta (o proceso) contiene:

```
root@unai-server:/proc/2554# ls
attr coredump_filter gid_map mountinfo oom_score sched stat uid_map
autogroup cpuset io mounts oom_score_adj schedstat statm wchan
auxv cwd limits mountstats pagemap sessionid status
cgroup environ loginuid net patch_state setgroups syscall
clear_refs exe map_files ns personality smaps task
cmdline fd maps numa_maps projid_map smaps_rollback timers
comm fdinfo mem oom_adj root stack timerslack_ns
```

- fd → Listado de ficheros abiertos por el proceso
- stat → Estado del proceso: PID, PPID, utime, ...

COMANDOS PARA GESTIONAR PROCESOS

- top → Estado del sistema (carga de CPU, memoria,...)
 - Tiene comandos propios controlar su uso (q para salir)
 - Shift f para entrar a panel de orden
- ps → Información sobre los procesos activos
 - Tiene parámetros para recuperar información
 - P.e. ps aux muestra estadísticas por proceso
- kill → Enviar una señal a un proceso
 - P.e. kill -9 miProceso fuerza su terminación
- ps tree → Árbol de procesos

OTROS COMANDOS

- sed (Stream Editor) → Modificar un fichero dado como entrada
 - La modificación se hace línea a línea
 - Sintaxis: sed <opciones> <instrucciones> <fichero>
 - Opciones:
 - -i → El fichero de entrada es sobrescrito
 - Instrucciones:
 - i → Insertar línea antes de la actual
 - p → Mostrar línea actual en salida estándar
 - s → Reemplazar patrón por otro patrón en línea actual
 - Ejemplos:
 - sed -i 's/casa/coche/g' a.txt → Reemplazar "casa" por "coche" en a.txt
 - sed -i '/cadena/d' archivo → Eliminar toda ocurrencia de cadena en a.txt
 - sed '2,3 p' * → Mostrar líneas 2 y 3 de todos los ficheros
- Awk → Lenguaje de programación diseñado para procesado de texto
 - Sintaxis general de un programa: patrón { acción }
 - Patrón → Cuándo aplicar la acción
 - Procesa el texto línea a línea
 - Si el patrón es cierto, se aplica la acción para esa línea
 - Si no se proporciona patrón, se aplica la acción a todas las líneas
 - Ejemplos:
 - awk '{print \$1,\$4}' data.txt → Muestra la 1ª y 4ª columna de data.txt
 - awk '{print \$1,\$NF}' data.txt → Muestra la 1ª y última columna de data.txt
 - awk 'END { print NR }' data.txt → Cuenta el nº de líneas de data.txt