

# Administración Linux

## Administración de Sistemas

Unai Lopez Novoa  
unai.lopez@ehu.eus

eman ta zabal zazu



Universidad  
del País Vasco

Euskal Herriko  
Unibertsitatea

# Intérprete de comandos (Shell)

- Interfaz del sistema entre usuario/aplicaciones y las llamadas al sistema operativo
- Es un programa del sistema operativo con los mismos privilegios que cualquier otro
- Símbolo de uso:
  - \$ usuario normal
  - # superusuario / administrador / root
- Tipos de acceso:
  - Local: Terminales de texto (Ctrl+Alt+F1...6) y una terminal gráfica (Ctrl+Alt+F7)
  - Remoto: A través de la red: telnet, rlogin, ssh, ...



# Intérprete de comandos (Shell)

- Múltiples Shell en los sistemas Unix/Linux, p.e.:

|      |   |
|------|---|
| 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<br><i>zsh es la shell por defecto desde macOS 10.15</i> |

- Es importante conocer cuál estamos usando



# Intérprete de comandos

- Comando **man**
  - Debería ser el primer comando a aprender
  - Muestra información sobre un comando, función, ...
- Sintaxis: `man <comando>`
- Los manuales se encuentran en `/usr/share/man`
  - Normalmente se busca alfabéticamente y muestra la 1ª entrada que coincida
- Configuración en `/etc/manpath.config`



# Sistema de ficheros

- Comando ls: listar archivos
  - Uno de los más utilizados
  - Algunos parámetros (combinables entre sí)
    - l           Mostrar información adicional de cada archivo (fecha, tamaño, permisos, ...)
    - a           Mostrar todos los archivos, incluidos los ocultos
    - r           Listar en orden inverso
    - t           Listar en orden cronológico ascendente
    - h           Mostrar tamaños de ficheros en KB o MB en vez de bytes
- P.e. ls -larth muestra toda la información de los archivos de un directorio ordenados de manera cronológica y con tamaños en KB o MB



# Sistema de ficheros

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



# Sistema de ficheros

- 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 una archivo  
*Ejemplo: grep casa miArchivo.txt*
- cut Muestra secciones concretas de un archivo  
*Ejemplo: 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



# Usuarios

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

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

The diagram illustrates the fields of the line `unai:x:1011:1011:Unai Lopez Novoa:/home/unai:/bin/bash` with arrows pointing to their descriptions:

- unai: Nombre de usuario
- x: Contraseña cifrada en `/etc/shadow`
- 1011: UID: ID de usuario
- 1011: GID: ID de grupo
- Unai Lopez Novoa: Nombre completo
- /home/unai: Directorio raíz
- /bin/bash: Shell de inicio

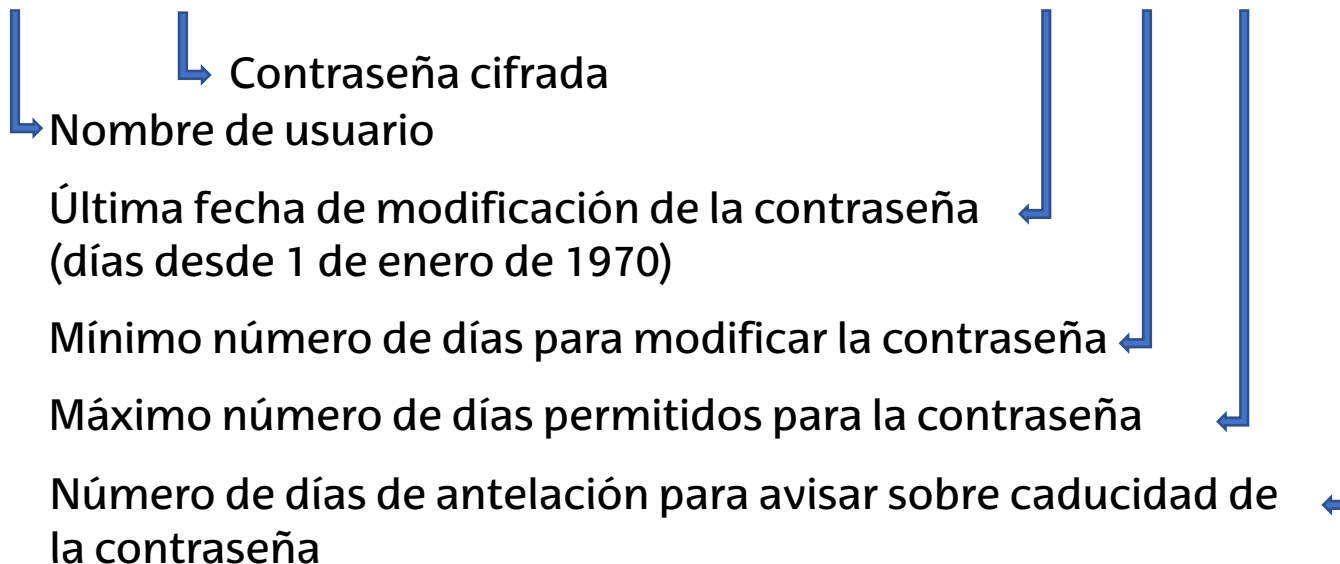




# Usuarios

- El fichero /etc/shadow contiene contraseñas del sistema
  - Hace falta permisos root para leerlo
- Cada línea tiene la siguiente forma:

```
unai:$MZdY3ECAD6tFmEtYAIOSdnqidnqwdibb1:18148:0:99999:7:::
```



# Usuarios

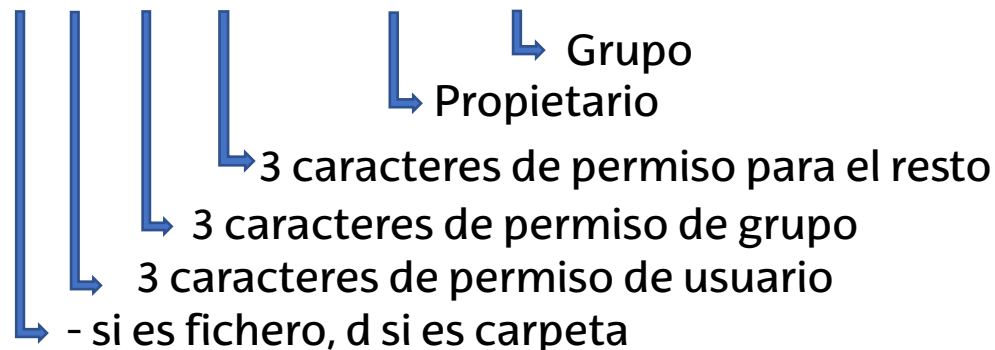
- Unix gestiona la seguridad del sistema de ficheros con usuarios, grupos y permisos para cada uno.
- Cada fichero (y carpeta) tiene un único propietario y unos permisos de acceso.
- Los permisos son:
  - r           Lectura: permite leer el contenido
  - w           Escritura: permite modificar el contenido
  - x           Ejecución: permite la ejecución de un fichero
- Cada permiso se configura para:
  - usuario:   El propietario del fichero
  - grupo:     Usuarios del grupo al que pertenezca el propietario
  - resto:      Resto de usuarios del sistema



# Usuarios

- Generalmente, los usuarios sólo tienen permisos de escritura en su directorio raíz (p.e. /home/<usuario>)
  - Y en algunos directorios temporales como /tmp
- El super-usuario (o sysadmin) tiene acceso todo el sistema de archivos
- Ejemplo de permisos de un archivo (con ls -l):

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



# Usuarios

- Comandos básicos de gestión de usuarios

|         |  |
|---------|--|
| whoami  | Muestra el nombre del usuario actual       |
| who     | Muestra los usuarios conectados al sistema |
| w       | Equivalente a who, algo más de información |
| 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            |



# Ejercicios 1

- Crear un fichero Ej1.txt que contenga la frase “Hola EHU”.
  - Modificad sus permisos para que vuestro usuario no pueda leerlo.
- Contar el número de líneas del fichero .bashrc en vuestro directorio raíz de usuario
- Mostrar los nombres de los usuarios del sistema
  - Pista: 1ª columna del fichero /etc/passwd
- Mostrar la información de los usuarios que tengan bash como Shell de inicio
  - Pista: esta información también está en /etc/passwd
- Crear un enlace simbólico a /bin/bash en /tmp llamado “miBash”



# Variables de entorno

- Sirven para guardar valores en una sesión de Shell
- Para leer el contenido: `echo $VARIABLE`
  - P.e. podéis probar a ejecutar `echo $HOME`
- Hay de 2 tipos:
  - Del usuario:
    - Se mantienen durante la sesión activa, después se eliminan
    - Se listan con el comando **env**
    - P.e. `$HOME`, `$LD_LIBRARY_PATH`
  - Del sistema:
    - Están en todas las sesiones del sistema
    - Se listan con el comando **set**
    - P.e., `$BASH`, `$HOSTNAME`



# Variables de entorno

- Depende de la Shell, las variables de entorno se crean de manera diferente
- En una Shell tipo sh
  - P.e. Bash, ksh
  - Comando **export**, p.e. export MIVARIABLE=4
- En una Shell tipo csh
  - P.e. tcsh, zsh
  - Comando **setenv**, p.e. setenv MIVARIABLE 4
- Estas variables desaparecen al finalizar la sesión



# Variables de entorno

- Algunas 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   |





# Variables de entorno

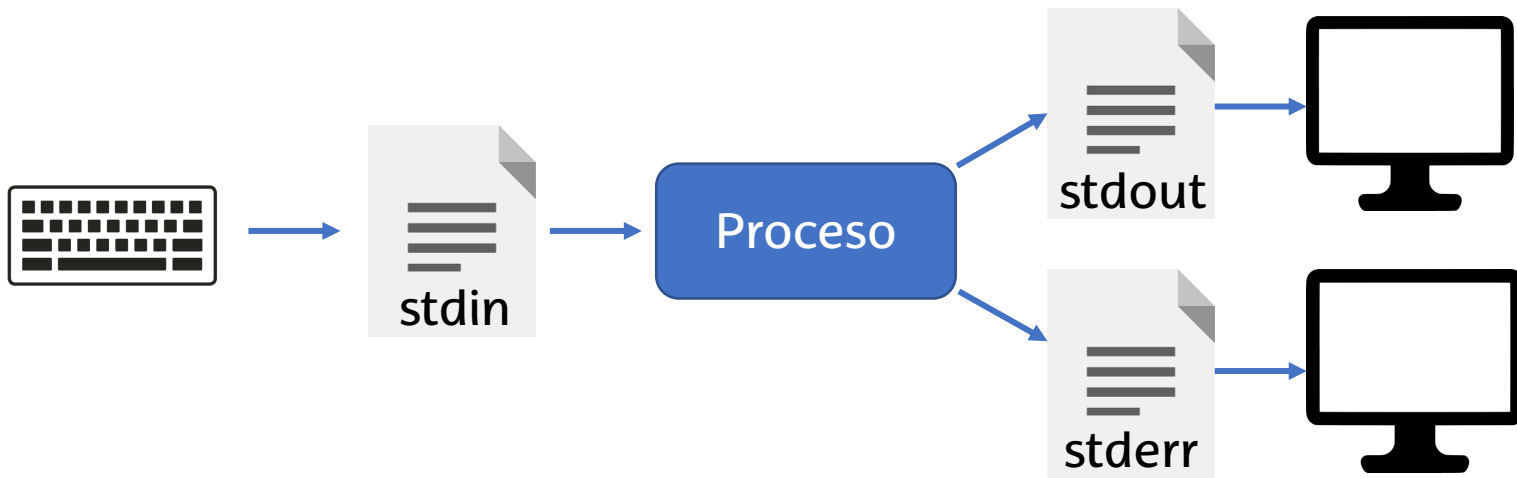
- Hay ficheros del sistema que permiten configurar la Shell
- Podemos definir variables de entorno en ellos para que estén siempre disponibles en nuestras sesiones
- En Bash, en orden de relevancia:  
    /etc/bashrc → /etc/profile → \$HOME/.bashrc → \$HOME/.bash\_profile
- En zsh:  
    /etc/zshrc → /etc/profile → \$HOME/.zshrc → \$HOME/.zprofile



# Redirección y Pipes

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

- `stdin` Teclado (entrada estándar)
- `stdout` Pantalla (salida estándar)
- `stderr` Pantalla (salida de errores)



# Redirección y Pipes

- **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 y Pipes

- 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



# Redirección y Pipes

- 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”
- Además, en Bash se puede **concatenar** la ejecución de varios comandos seguidos
  - Ejemplo: `ls -l; cd ..; ls -l`
  - Alternativamente: `ls -l && cd .. && ls -l`



# Ejercicios 2

- Mostrar el usuario del sistema que cambió la contraseña hace más tiempo
  - Pista: la información está en /etc/shadow
- Crear una variable de entorno CIUDAD persistente con el nombre de una ciudad.
  - Cerrar y volver a abrir sesión en el sistema, comprobar que existe.
- Guardar en un fichero las 4 últimas columnas resultantes de llamar al comando "ping www.google.es"
  - Utilizar el parámetro "-c" de ping para limitar el número de iteraciones.



# Shell scripting

- Un guion de Shell (Shell script) es un conjunto de comandos que se ejecutan con un objetivo concreto.
- Su forma más simple: un fichero de texto con un comando por línea.

- Ejemplo:

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

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

- Para ejecutar un script:
  - Si el script no tiene permisos: *bash miScript*
  - Alternativamente, modificar los permisos del script y ejecutarlo como un binario: *chmod +x miScript; ./miScript*



# Shell scripting

- Entrada/salida en un script

read                      Para leer del teclado

echo/printf            Para escribir por pantalla (stdout)

- Un script puede recibir parámetros

- Los parámetros que reciba se convierten 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 + ...)





# Shell scripting

- Control de flujo

- Sentencia if - else

- en /bin/bash      `if [[ <condición> && <condición> ]]; then`
    - en /bin/sh      `if [ <condición> ] && [ <condición> ]; then`

- Ejemplo

```
#!/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"



# Shell scripting

- Control de flujo
  - Operadores de comparación

| Strings               | Numéricos   | Verdadero si   |
|-----------------------|---|--|
| <code>x = y</code>    | <code>x -eq y</code>                                      | x es igual a y   |
| <code>x != y</code>   | <code>x -ne y</code>                                      | x es diferente a y   |
| <code>x &lt; y</code> | <code>x -lt y</code> *                                    | x es menor que y   |
|                       | *: <code>-gt</code> , <code>-le</code> , <code>-ge</code> | equivalen: <code>&gt;</code> , <code>&lt;=</code> , <code>&gt;=</code> |

| Operador   | Verdadero si   |
|--|--|
| <code>-d carpeta</code>                            | Carpeta existe   |
| <code>-e fichero</code>                            | Fichero existe   |
| <code>-r fichero</code><br><code>-w fichero</code> | Usuario tiene permiso de lectura / escritura en el fichero |
| <code>-s fichero</code>                            | Fichero existe y no está vacío                             |

- Sentencia **case**

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



# Shell scripting

- Bucles

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

- 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
```



# Shell scripting

- Variables

- Todas las variables en Bash se consideran Strings
- Se puede hacer aritmética con variables Bash

- Utilizar (( ))
- Ejemplo:

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

Concatenación de strings

Suma aritmética

- Se pueden crear listas de variables

- Utilizar list
- Ejemplo:

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

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

Muestra todos los elementos

Muestra el 1er elemento



# Shell scripting

- Expresiones regulares

- Utilizadas 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"
  - "casa", "cbsa", "ccsa",...
- "c([a-z]+)sa"
  - "casa", "caasa", "cxyzsa"
- "c(\d\*)sa"
  - "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          |

- Un listado exhaustivo de caracteres especiales:

<https://cheatography.com/davechild/cheat-sheets/regular-expressions/>



# Shell scripting

- Funciones

- Se pueden crear funciones para ordenar el código
- Se le pueden pasar parámetros
  - Se recogen en la función como \$1, \$2, ... en base al orden
- Para devolver un valor de la función, utilizar **echo** y no return
  - return se interpreta como el comando de finalizar el programa

- Ejemplo:

```
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"
```



# Ejercicios 3

- Hacer un script que muestre los nombres de todos los archivos en /tmp usando un bucle
- Extender el script anterior haciendo que, si /tmp está vacío, cree un archivo llamado vacio.txt en /tmp
- Hacer un script que implemente una calculadora
  - El script debe recibir 2 parámetros numéricos
  - El script debe mostrar una lista de las siguientes operaciones y pedir al usuario que introduzca la operación deseada
    - Suma, resta, multiplicación, división
  - Cada operación debe implementarse como una función



# Bibliografía

- Este tema está basado en:
- Pablo Abad Fidalgo, José Ángel Herrero Velasco. Advanced Linux System Administration, Topic 2: Command Line (Shell). OCW UNICAN 2018.
  - Publicado bajo licencia Creative Commons BY-NC-SA 4.0
  - <https://ocw.unican.es/course/view.php?id=241>
- Consultado en junio 2020

