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 *zsh es la shell por defecto desde macOs 10.15*

- 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

- Jerarquía habitual en sistemas Unix o Linux

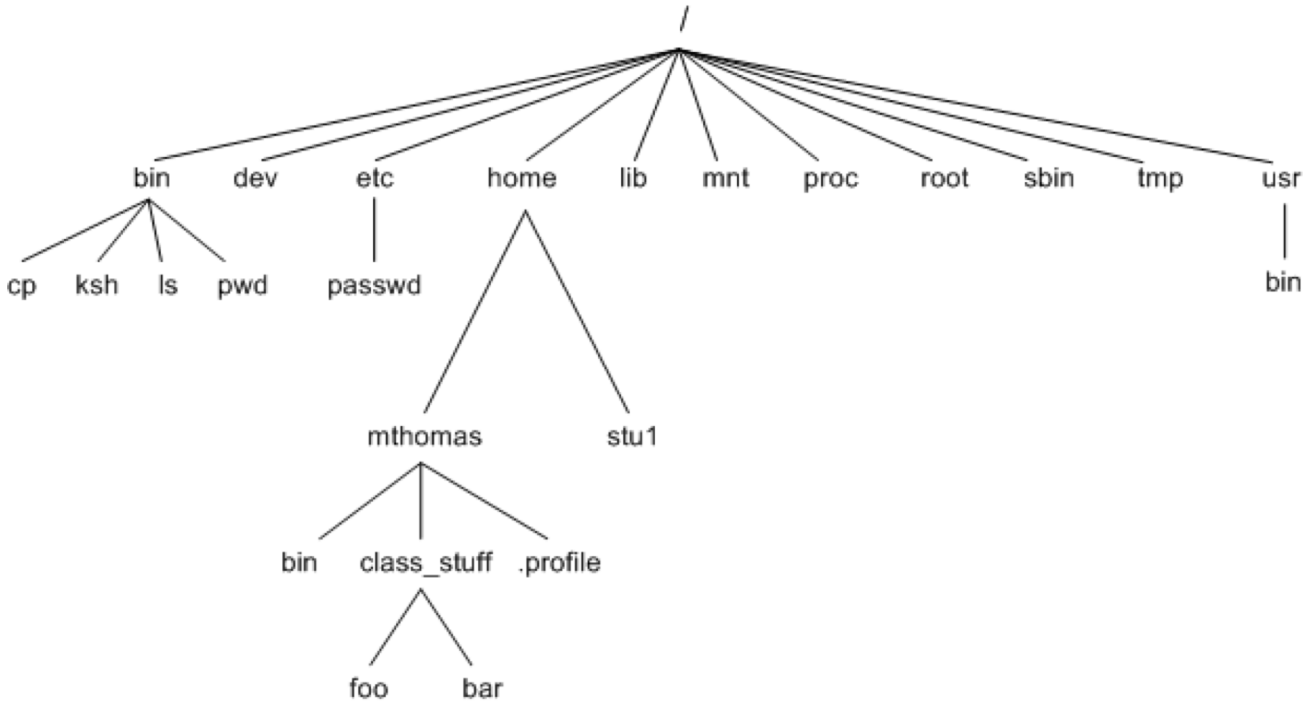


Imagen de: https://homepages.uc.edu/~thomam/Intro_Unix_Text/Images/unix_file_system.png

Sistema de ficheros

- Estructura de árbol
- Formas de acceso:
  - Absolutas: /home/unai/miArchivo.txt
  - Relativas a la ruta actual: ../../miCarpeta/miArchivo.txt
- Los archivos ocultos comienzan su nombre con "."
- Unidades externas (CDs, HDDs externos, ...):
  - Se pueden asociar a cualquier posición de la jerarquía
  - Un mismo programa puede tratar ficheros internos y dispositivos externos de manera indiferente

Sistema de ficheros

/	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 operativos.
/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.

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

Atajos

- Autocompletado de Shell
  - Utilizar la tecla tabulador para autocompletar nombres/rutas
  - Si hay varias posibles opciones, las muestra antes de completar
- Expresiones regulares con caracteres comodín (wildcards)
  - Algunos caracteres se usan con fines especiales:
    - `*` Reemplazar todos los caracteres
    - `?` Reemplazar un único caracter
    - `[]` Reemplazar por un rango númerico, p.e., [12]
    - `~` Atajo para el directorio raíz del usuario ($HOME)
  - Si hiciera falta buscar un carácter de estos en un archivo, escribirlo precedido por \ o rodeado por “”

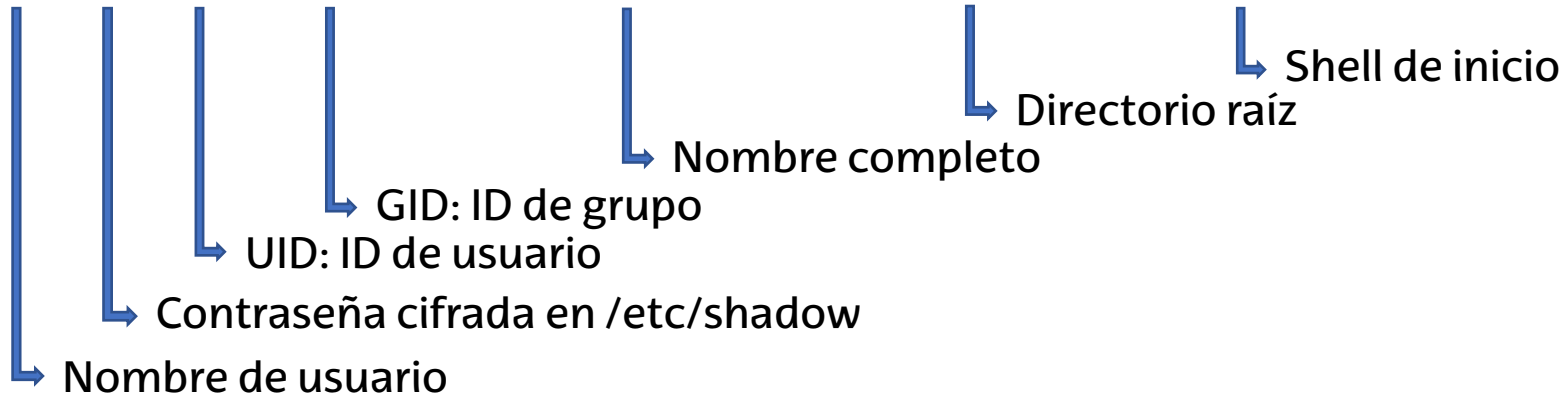
Atajos: historia de comandos

- La Shell guarda el histórico de comandos que hemos tecleado, se puede visualizar con **history**
  - Una vez mostrado el histórico, se puede volver a ejecutar un comando escribiendo !‹nº de comando›
  - P.e. si el nº 20 ha sido **ls**, escribir !20 ejecutará ls de nuevo
  - !! ejecuta de nuevo el último comando
- Con los cursores arriba/abajo se navega por la historia más reciente
- Pulsar Ctrl+r y empezar a escribir para que la Shell autocomplete comandos históricos
  - Utilizar Ctrl+r en este modo para navegar por las opciones

Usuarios

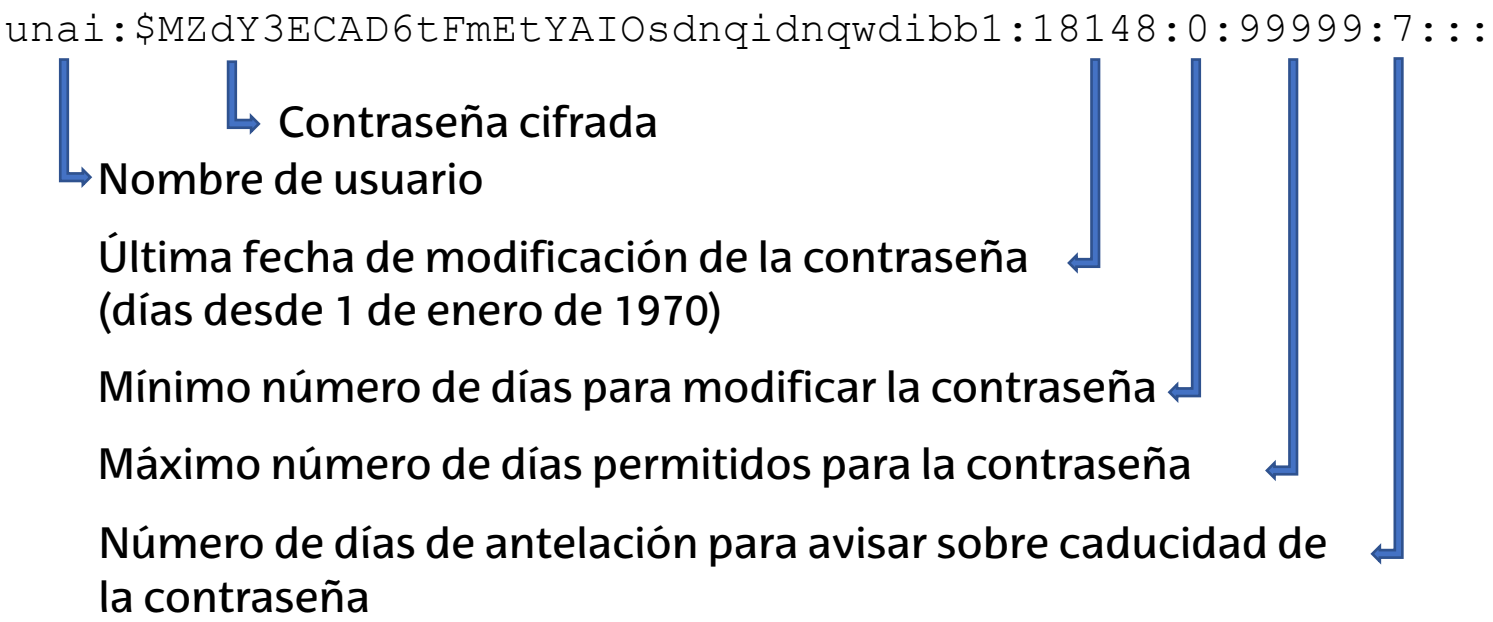
- En Unix, los usuarios están organizados en grupos
- Los ficheros /etc/passwd y /etc/group contienen la información sobre los usuarios y grupos del sistema
  - Cada línea tiene diferentes campos separados por el carácter :
  - Ejemplo de línea:

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



Usuarios

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



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

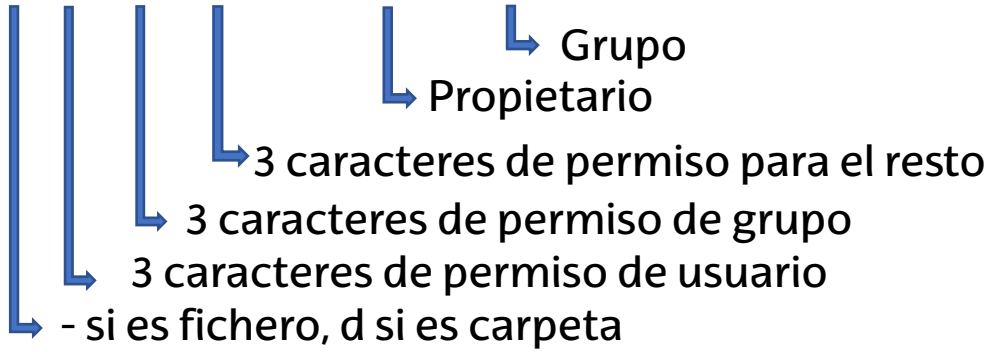
- Cada permiso se configura para:

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

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 unos ficheros concretos en el sistema que permiten configurar la Shell
- Podemos definir variables de entorno en ellos para que estén siempre disponibles en nuestras sesiones

- En Bash hay varios, en orden de relevancia:

  /etc/bashrc → /etc/profile→$HOME/.bashrc→$HOME/.bash_profile

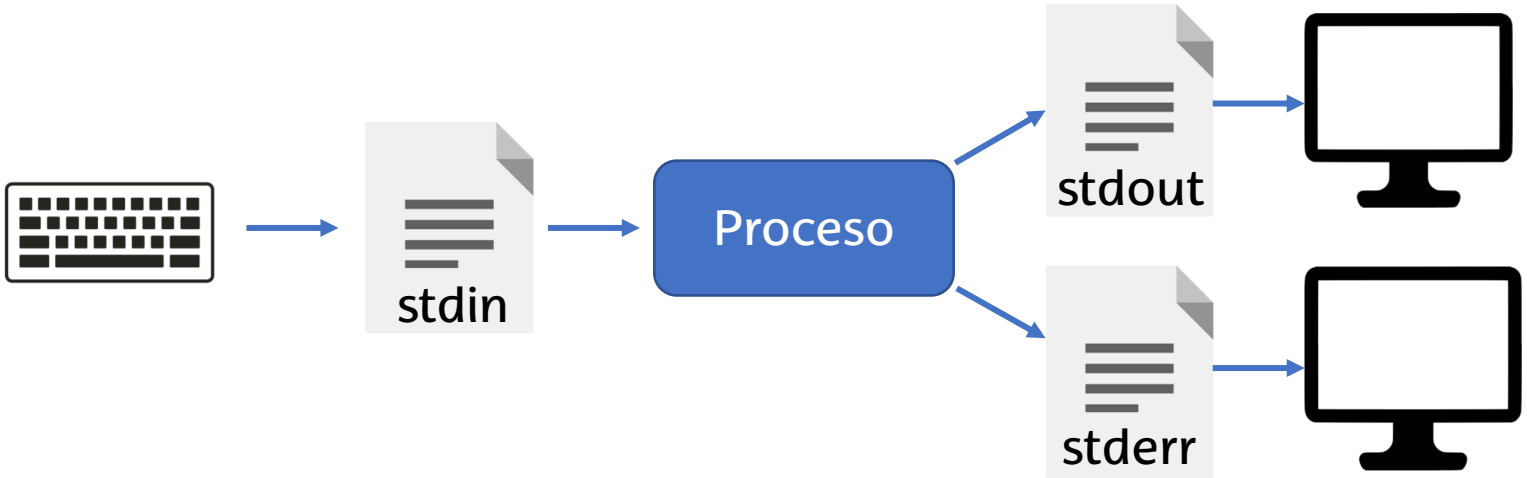
- Igualmente en zsh:

  /etc/zshrc → /etc/profile→$HOME/.zshrc→$HOME/.zprofile

Redirección y Pipes

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 1

- 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"
- Contar el número de líneas del fichero .bashrc en vuestro directorio raíz de usuario

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.ehu.eus”
  - 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

Comillas `` para ejecutar un comando

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

Atención al “fi” para cerrar el bloque “if”

Shell scripting

- Control de flujo
  - Operadores de comparación

Strings	Numéricos	Verdadero si
x = y	x -eq y	x es igual a y
x != y	x -ne y	x es diferente a y
x ‹ y	x -lt y *	x es menor que y
	*: -gt, -le, -ge	equivalen: › , ‹= , ›=

Operador	Verdadero si
-d carpeta	Carpeta existe
-e fichero	Fichero existe
-r fichero -w fichero	Usuario tiene permiso de lectura / escritura en el fichero
-s fichero	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:

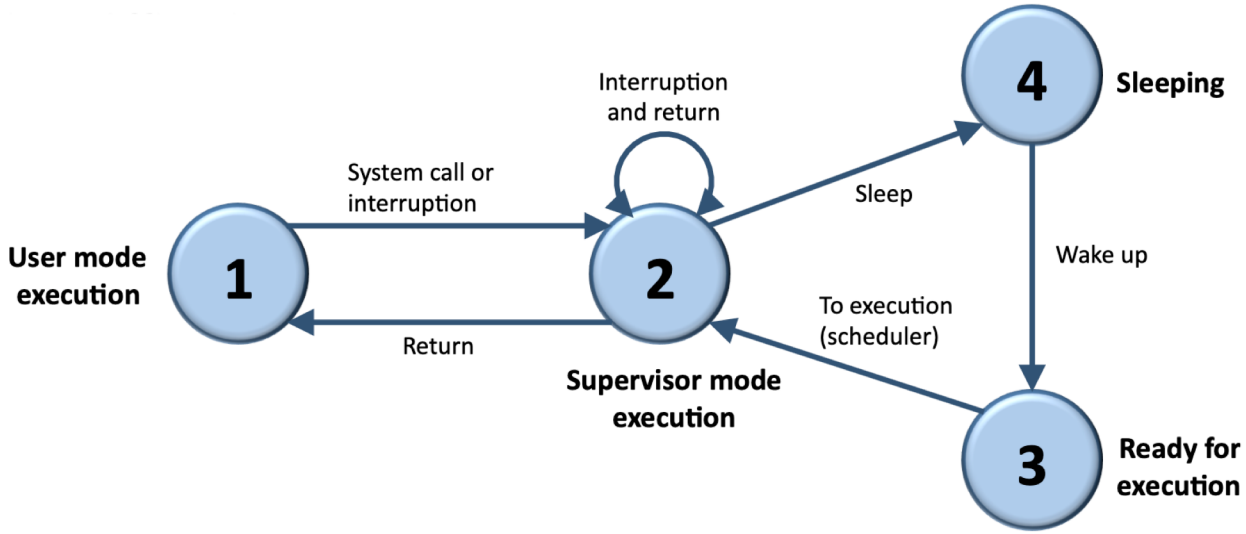
```
function suma()
{
  OP1=$1
  OP2=$2
  echo $(($OP1+$OP2))
}

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

Gestión de procesos

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



Gestión de procesos

- En Unix, los procesos 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
- En cada sesión, 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

Gestión de procesos

- Hasta ahora, hemos lanzado todo en primer plano:
  - Ejemplo: ./miScriptLargo
- Para lanzar un proceso en 2º plano, añadir & al final
  - Ejemplo: ./miScriptLargo **&**
- Por otra parte, el comando **nohup** mantiene un proceso vivo aunque se cierre la sesión
  - Nohup significa *No Hang Up*
- Al combinar nohup y &, podemos dejar un proceso en ejecución y cerrar nuestra sesión de Shell
  - Ejemplo: **nohup** ./miScriptLargo **&**

Gestión de procesos

- En la carpeta /proc se encuentra la información asociada a los procesos, que es usada por el kernel
  - Cada proceso tiene una carpeta, por ejemplo:

```
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_rollup 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, ...

Gestión de procesos

- Comandos para gestionar procesos
  - top Muestra el estado del sistema (carga de CPU, memoria,...)
    Tiene comandos propios controlar su uso (q para salir)
    P.e, Shift+M ordenar por consumo de RAM
  - ps Muestra 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
  - pstree Muestra el árbol de procesos

Otros comandos útiles

- sed (Stream Editor)
  - Comando para 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, en lugar de usar stdout
    - 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

Otros comandos útiles

- Awk
  - Es un lenguaje de programación diseñado para procesado de texto
  - Su nombre deriva de los apellidos de sus creadores
    - Alfred **A**ho, Peter **W**einberger, Brian **K**ernighan.
  - Sintaxis general de un programa awk: patrón { acción }
    - El patrón determina cuando 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 aplicación 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 columa de data.txt
awk 'END { print NR }' data.txt	Cuenta el nº de líneas de data.txt

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