

Introducción al curso y al programa

Curso: Desarrollo de servicios en la nube con HTML5, Javascript y node.js

Curso de **nivel medio**
de **programación** en **JavaScript**
que introduce **ingeniería de software**
para desarrollar **aplicaciones de servidor**
basadas en **node.js** y **express.js**
accesibles con **HTML5** desde
terminales fijos y móviles



Diseño de servicios en la nube, utilizando JavaScript, para acceso móvil y multi-dispositivo con HTML5*

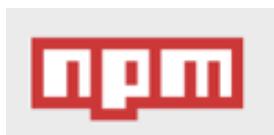


◆ Programa de especialización con 5 cursos MOOC

- Desarrollo de Aplicaciones en HTML5, CSS y Javascript, incluyendo Dispositivos Móviles Firefox O.S.
- Desarrollo avanzado de Aplicaciones HTML5 y Firefox O.S.* , incluyendo técnicas de ingeniería de software
- Desarrollo de aplicaciones HTML5 multi-terminal (Android, iOS, ...)*
- **Desarrollo de servicios en la nube con HTML5, Javascript y node.js***
- Desarrollo avanzado de servicios en la nube con Javascript y node.js*



*Nota: Esperamos que estos cursos empiecen en MiriadaX en 2015 y 2016.



Express



- ◆ Modulo 0. Introducción al curso y al Sistema Operativo UNIX
- ◆ Modulo 1. Introducción a JavaScript de servidor y a node.js. Sentencias, Variables, Booleanos, Números, Strings y Funciones.
- ◆ Modulo 2. Introducción a JavaScript de servidor y a node.js. Bucles, Clases predefinidas, Objetos, Propiedades y Métodos; Prototipos y Clases; Arrays; JSON; Funciones como Objetos y Cierres (Closures).
- ◆ Modulo 3. Modulos node.js; Expresiones Regulares; Eventos, Entorno de Ejecución y Conurrencia en node.js; Ficheros y Flujos.
- ◆ Modulo 4. Introducción a HTTP y a los Servidores Web; Introducción a express y al Middleware Static; Introducción a REST; Aplicaciones express.js y Composición de Middlewares; Formularios GET y POST; Parámetros Ocultos.
- ◆ Modulo 5. Gestión de versiones de proyectos con git y GITHUB; Proyecto, Espacio de Trabajo y Versiones (Commit); Arboles y Ramas de un proyecto; Repositorios Remoto y colaboración a través de GITHUB.
- ◆ Modulo 6. Proyecto Quiz I: Patrón Modelo-Vista-Controlador (MVC); generación del proyecto con express-generator; Primera Página y Primera Pregunta; Despliegue en la nube (Heroku).
- ◆ Modulo 7. Proyecto Quiz II: La Base de Datos (DB), Tablas, sequelize.js y SQLite; Despliegue en Heroku utilizando Postgres; Presentación de Listas de Quizes y Autoload.
- ◆ Modulo 8. Proyecto Quiz III: Gestión de Listas de Quizes, Creación, Edición y Borrado.
- ◆ Modulo 9. Proyecto Quiz IV: Creación y Moderación de Comentarios a Quizes; Relaciones entre Tablas de la Base de Datos; Sesiones, Autenticación y Autorización; HTTP Seguro (HTTPS).



Módulos

Desarrollo y evaluación del curso

◆ El curso consta de 5 tramos de 2 modulos (salvo tramo 5)

- Diseñado para realizarse en 5 semanas (1 semana por tramo)
 - ◆ Pero se deja una semana adicional por tramo
 - ◆ tiempo máximo de realización ~12 semanas (2 semanas por tramo)

◆ Ejercicios P2P de **entrega obligatoria**

- Al final de cada módulo par (1 ejercicio por tramo)
 - además hay tests obligatorios y mas ejercicios P2P opcionales

◆ Apertura y cierre de tramos

- Tramo 1: comienzo del curso y cierra aprox. final semana 3
- Tramo 2: comienzo semana 2 y cierra aprox. final semana 5
- Tramo 3: comienzo semana 3 y cierra aprox. final semana 7
- Tramo 4: comienzo semana 4 y cierra aprox. final semana 9
- Tramo 5: comienzo semana 5 y cierra aprox. final semana 11

Actividades de un Módulo

◆ **Tarea 0:** Descargar transparencias y ejemplos del módulo

- Fichero ZIP para descargar con
 - ◆ Transparencias en formato PDF

◆ **Tareas de Aprendizaje (varias):**

- un **video** o **screencast** del tema (3 y 14 minutos)
 - ◆ evaluado (no siempre) con un **test** o un **ejercicio P2P** opcional

◆ **Tarea final:** Ejercicio P2P final

- Es de entrega obligatoria en los módulos pares: 2, 4, 6, 8 y 9

Equipos, herramientas y servicios a utilizar

◆ Un PC o portatil de trabajo con su sistema operativo

- UNIX: Linux (Ubuntu, ..), Mac OS X (BSD UNIX),
 - ◆ Windows se puede utilizar en modo comando, pero el soporte es menos estándar

◆ Editor de textos que se use habitualmente

- Recomendados: ATOM (<https://atom.io/>) o Sublime (sublimetext.com)

◆ Navegador Firefox o Chrome

- Con sus entornos de desarrollo

◆ Webstorm: JavaScript IDE (<https://www.jetbrains.com/webstorm/>)

- Gratis para profesores y alumnos (<https://www.jetbrains.com/student/>), sino 99\$

◆ Cuenta en GITHUB (gratis)

- para compartir proyectos GIT en Internet (<https://github.com>)

◆ Cuenta en Heroku (gratis)

- para publicar proyectos en la nube (<https://www.heroku.com>)



Final del tema

Muchas gracias!





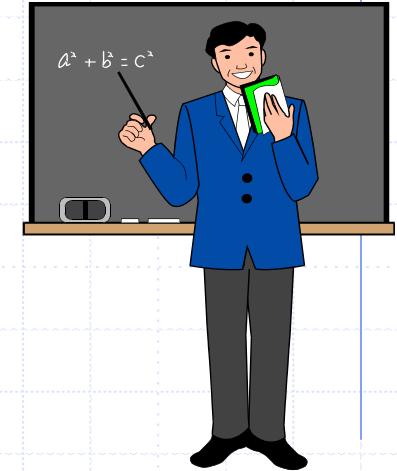
El sistema operativo UNIX

Introducción

Juan Carlos Yelmo

Contenidos

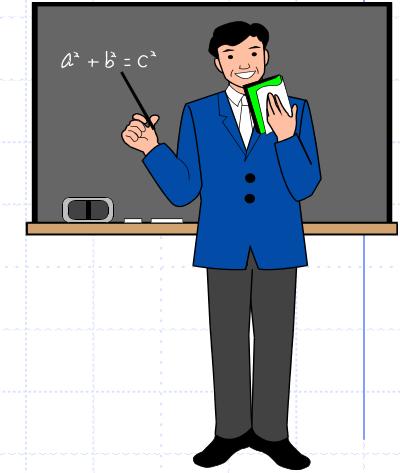
1. Introducción
2. El sistema de archivos
3. La interfaz de usuario



Contenidos

1. Introducción

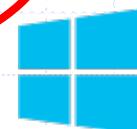
- Perspectiva histórica
- Principios generales de diseño
- Entrada al sistema
- Comandos básicos



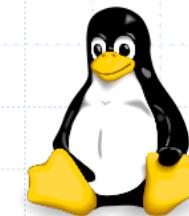
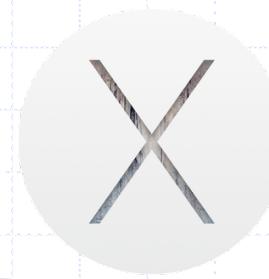
Introducción

◆ ¿Qué es un sistema operativo?

- Software básico de un computador que maneja la interfaz con el hardware, planifica tareas, asigna espacio de almacenamiento y presenta una interfaz de usuario por defecto cuando no está ejecutando ningún programa de aplicación



Windows



Introducción a UNIX

- ◆ Sistema operativo de propósito general, multiusuario y multitarea para todo tipo de máquinas y aplicaciones
- ◆ Diseñado por y para programadores
- ◆ Como entorno de programación, su contexto habitual de uso es el de un equipo de trabajo cooperando en el desarrollo de sistemas software complejos: trabajo conjunto e intercambio controlado de información

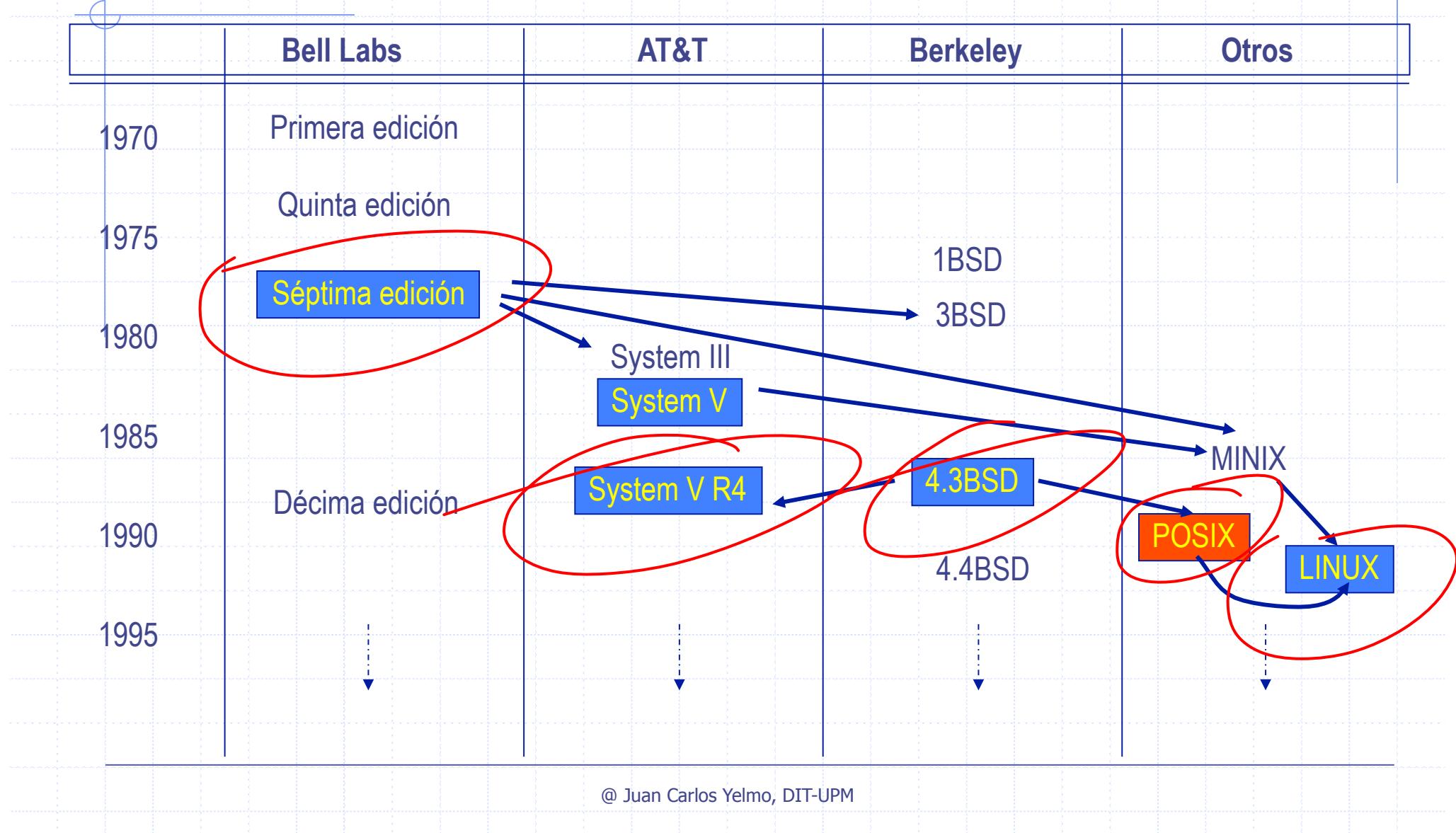
Historia



Kenneth L. Thompson Dennis M. Ritchie

- ◆ Desarrollado en 1969 para un DEC PDP-7 por Ken Thompson (Bell Labs)
- ◆ Reescrito en C en 1973. Lenguaje recién desarrollado por Dennis Ritchie
- ◆ Código fuente distribuido gratuitamente a universidades en 1974
- ◆ La Universidad de Berkeley mejoró notablemente el original dando lugar al Berkeley UNIX (BSD)

Historia



Arquitectura de UNIX



Entrada al sistema

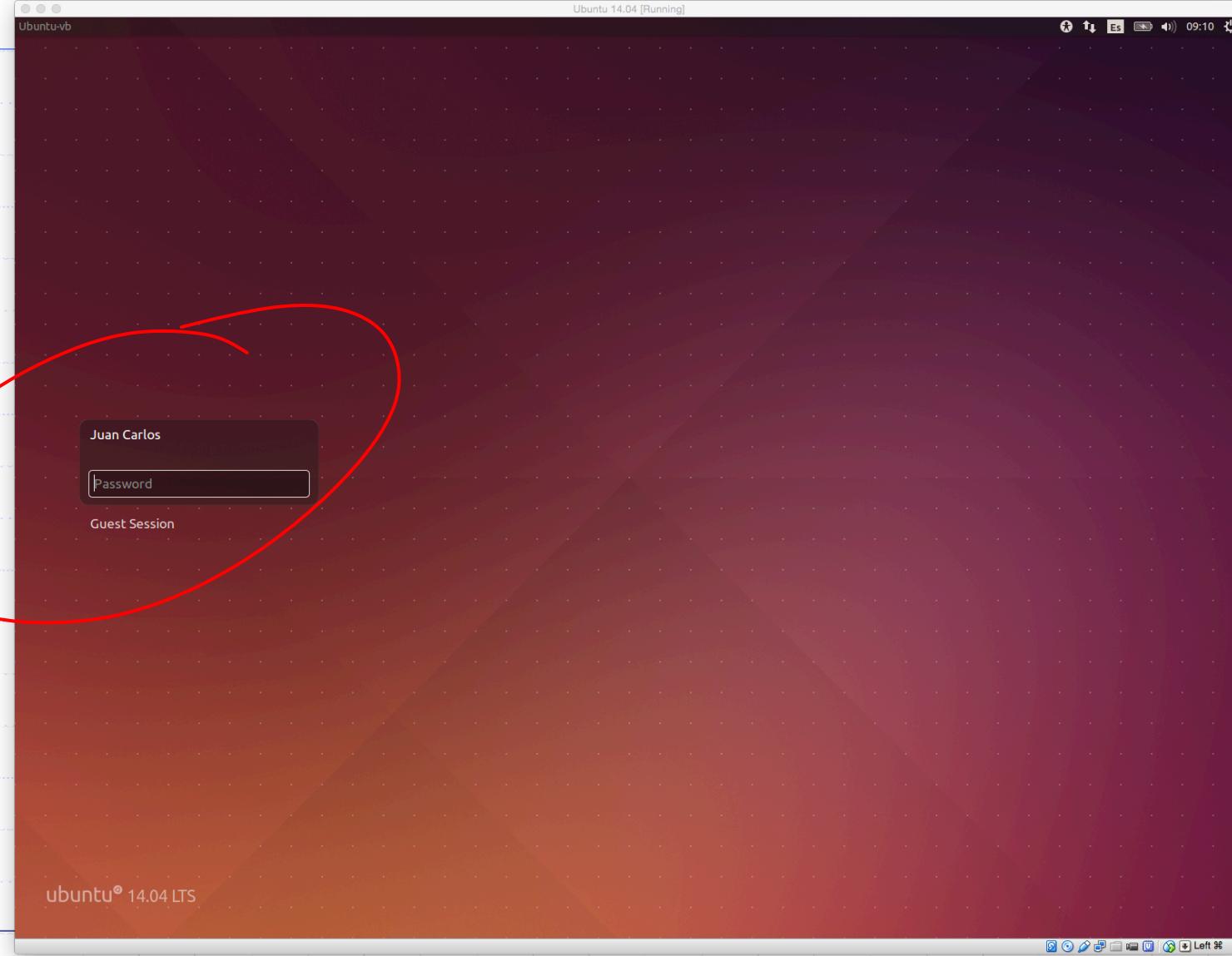
- ◆ Para utilizar UNIX es necesario identificarse y registrarse (*log in*) en el sistema, proporcionando un nombre (*user id*) y una contraseña (*password*)

```
login: jcyelmo
Password:
Last login: Mon Nov  5 16:35:13 CET 2012 on pts/2
Welcome to Ubuntu 12.04.1 LTS (GNU/Linux 3.2.0-32-generic x86_64)
```

* Documentation: <https://help.ubuntu.com/>

jcyelmo@jcyelmo-VirtualBox:~\$

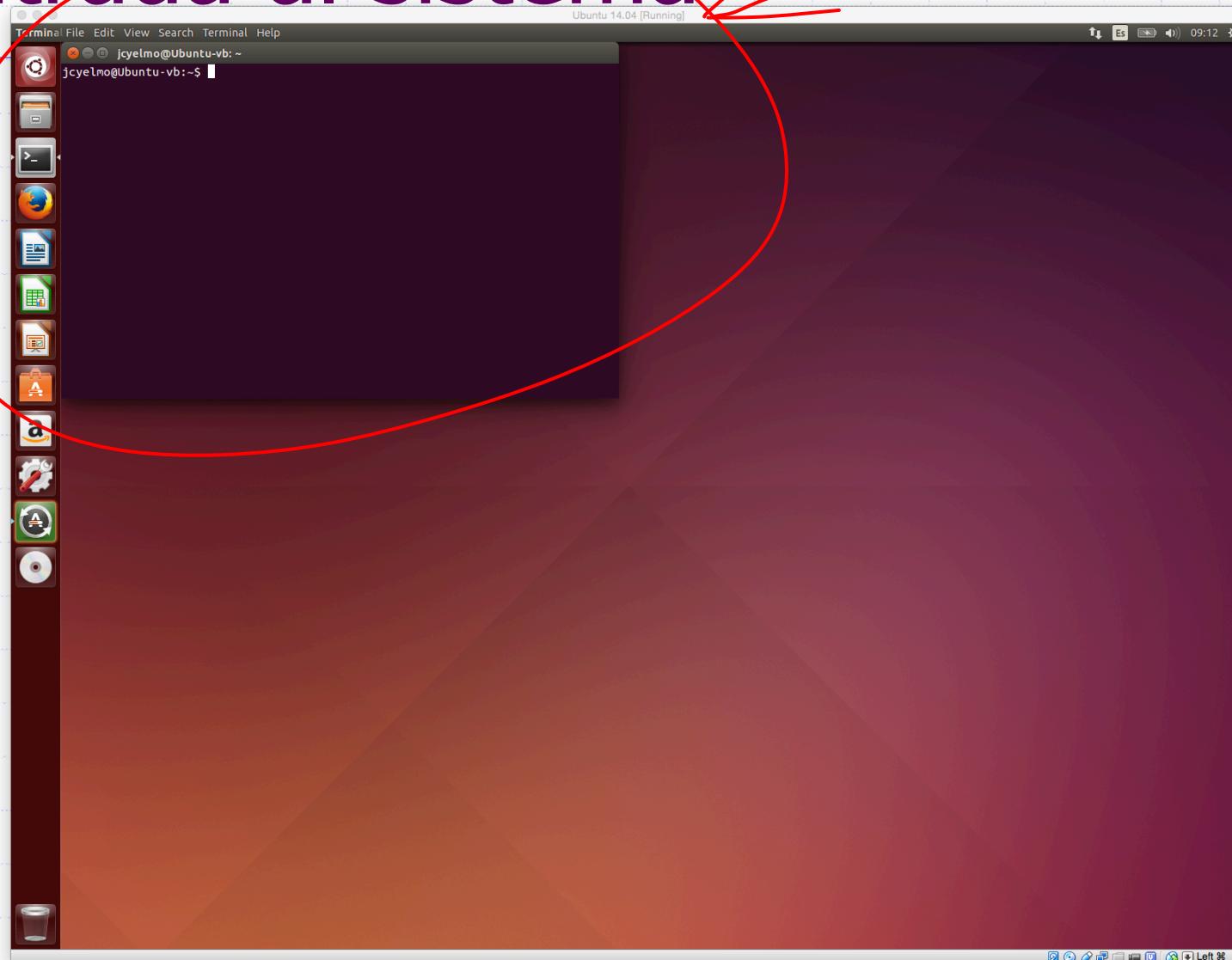
Entrada al sistema



@ Juan Carlos Yelmo, DIT-UPM

Entrada al sistema

terminal



Entrada al sistema

- ◆ UNIX mantiene la información básica de usuarios autorizados en un fichero público, /etc/passwd o /etc/shadow, incluyendo las contraseñas (cifradas) con el formato:
 - ◆ <usuario>:<password>:<uid>:<gid>:<nombre>:<home>:<shell>
- Ejemplo:
 - ◆ root:x:0:0:root:/root:/bin/bash
- ◆ Tras un registro correcto, el sistema invoca al intérprete de comandos (la shell) y da inicio a la sesión de usuario

Interfaz de línea de comandos

- ◆ La shell muestra su disposición a aceptar comandos del usuario mediante el *prompt* (\$, >, %, ...)
- ◆ Una vez introducida una línea, la shell lee la primera palabra de la línea de comandos, interpreta que es el nombre de un programa, lo busca y, si lo encuentra, lo ejecuta

Interfaz de línea de comandos

◆ Sintaxis de comandos de la shell de UNIX

\$ comando [argumentos] <RC>

Prompt

Programa/acción a ejecutar

Modificadores o datos de entrada

Comandos básicos

```
$ ls
Desktop  Downloads  Music  Public  Videos
Documents examples.desktop Pictures Templates
$
```

Dudas existenciales

¿Quién soy?

<code>whoami/who am i</code>	Identificador de usuario
<code>id</code>	Identificador de usuario y grupos

¿Dónde estoy?

<code>pwd</code>	Lugar en el sistema de archivos
<code>hostname</code>	Nombre de la máquina

¿Quién está conmigo?

<code>who/finger</code>	Quién está registrado
<code>w</code>	Quién está registrado y qué hace

¿Qué día es hoy?

<code>date</code>	Fecha y hora
<code>cal</code>	Calendario del mes

Dudas existenciales

```
x - jcyelmo@Ubuntu-vb: ~  
jcyelmo@Ubuntu-vb:~$ whoami  
jcyelmo  
jcyelmo@Ubuntu-vb:~$ pwd  
/home/jcyelmo  
jcyelmo@Ubuntu-vb:~$ date  
mar dic 16 09:46:37 CET 2014  
jcyelmo@Ubuntu-vb:~$ cal  
Diciembre 2014  
do lu ma mi ju vi sá  
 1  2  3  4  5  6  
 7  8  9  10 11 12 13  
14 15 16 17 18 19 20  
21 22 23 24 25 26 27  
28 29 30 31  
  
jcyelmo@Ubuntu-vb:~$ 
```

Manual on-line

man [opciones][[sección]temas]

Muestra información del manual de referencia sobre el tema solicitado (normalmente un comando)

Opciones principales

-k palabras clave

Muestra entradas del manual donde aparece alguna de las palabras clave

Ejemplos

man intro

Introducción general

man man

Información sobre el comando man

man -k socket

Comandos relacionados con sockets



El sistema operativo UNIX

Introducción

Fin del tema



El sistema operativo UNIX

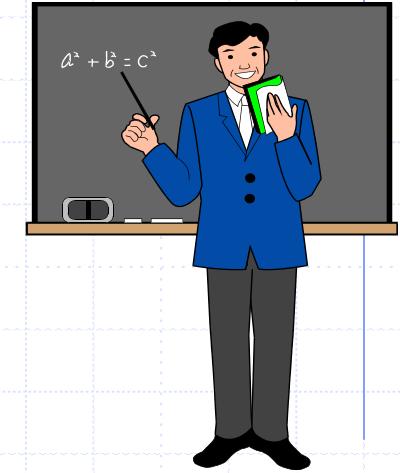
El sistema de archivos I

Juan Carlos Yelmo

Contenidos

2. El sistema de archivos

- Introducción
- Ficheros en UNIX
- Estructura del sistema de archivos
- Edición de ficheros



Introducción

- ◆ Los ordenadores pueden almacenar información de forma persistente en distintos medios físicos: cinta o disco magnético, disco óptico, discos SSD, etc.
- ◆ El sistema operativo presenta una visión lógica del almacenamiento de información que abstrae características de dispositivos físicos concretos y cuya unidad de almacenamiento es el *fichero* o archivo

Introducción

◆ Fichero

- Conjunto de información relacionada que se almacena en un dispositivo secundario (persistente) y a la cual se asigna un nombre
- Es la unidad lógica de almacenamiento secundario
- Normalmente es una secuencia simple de bytes de longitud finita

Introducción

◆ Sistema de archivos

- Mecanismo software que permite crear, almacenar, recuperar, proteger y gestionar ficheros
- Suele estar implementando como parte del núcleo (kernel) del sistema operativo y asocia ficheros con su implementación en los dispositivos de almacenamiento
- El sistema de archivos asocia a los ficheros información adicional como permisos de acceso, atributos, etc.

metadatos

Ficheros en UNIX

- ◆ Los archivos pueden contener textos, documentos, código fuente, ejecutables, directorios, páginas web, etc.
- ◆ Gran parte del sistema UNIX gira en torno al concepto de fichero. Se utilizan para representar también: dispositivos, buffers, sockets, etc.
- ◆ UNIX no impone estructura ni interpretación a la información contenida en un fichero. Éstas dependerán de las aplicaciones que lo utilizan.

Comandos básicos

Para listar ficheros

`ls`

Contenido de un directorio

Mostrar el contenido de un fichero

`cat`

Vuelca contenido a pantalla

`more`

Ambos muestran el contenido
página a página

`less`

Copiar, renombrar y mover ficheros

`cp`

Copia ficheros

`mv`

Mueve y renombra ficheros

`rm`

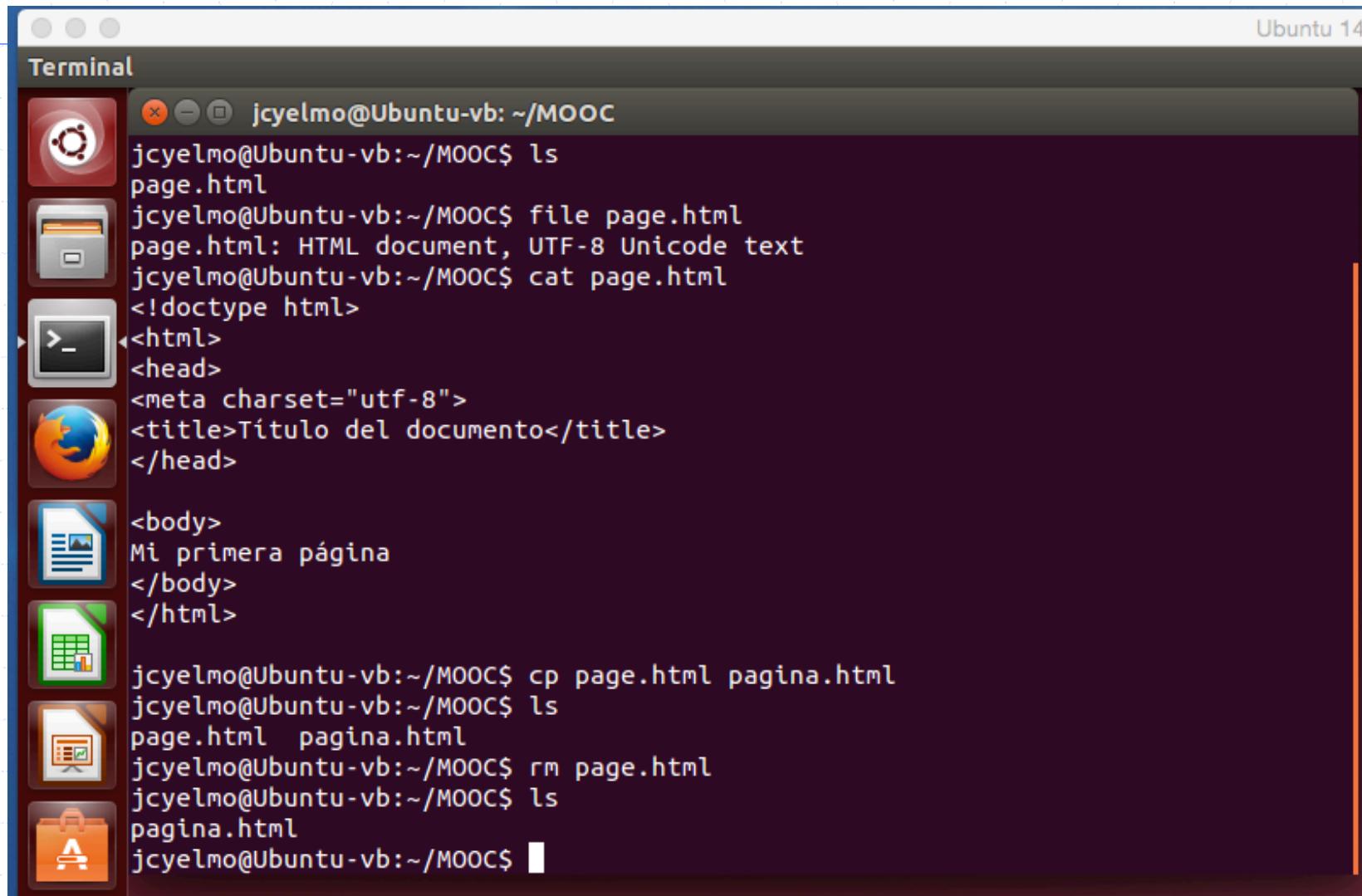
Elimina ficheros

Conocer el tipo de un fichero

`file`

Muestra el tipo de un fichero

Ficheros en UNIX



The image shows a screenshot of an Ubuntu 14.04 desktop environment. A terminal window titled "Terminal" is open, displaying a session of the Linux command-line interface. The session starts with the user logging in and navigating to their home directory (~). They then list files in the directory, identify a file as an HTML document, and view its contents. The user creates a copy of the file, lists the files again, and finally removes the original file. The terminal window has a dark background and light-colored text. The desktop environment includes icons for the Dash, Home, Applications, and Help, along with icons for the Dash, Home, Applications, and Help.

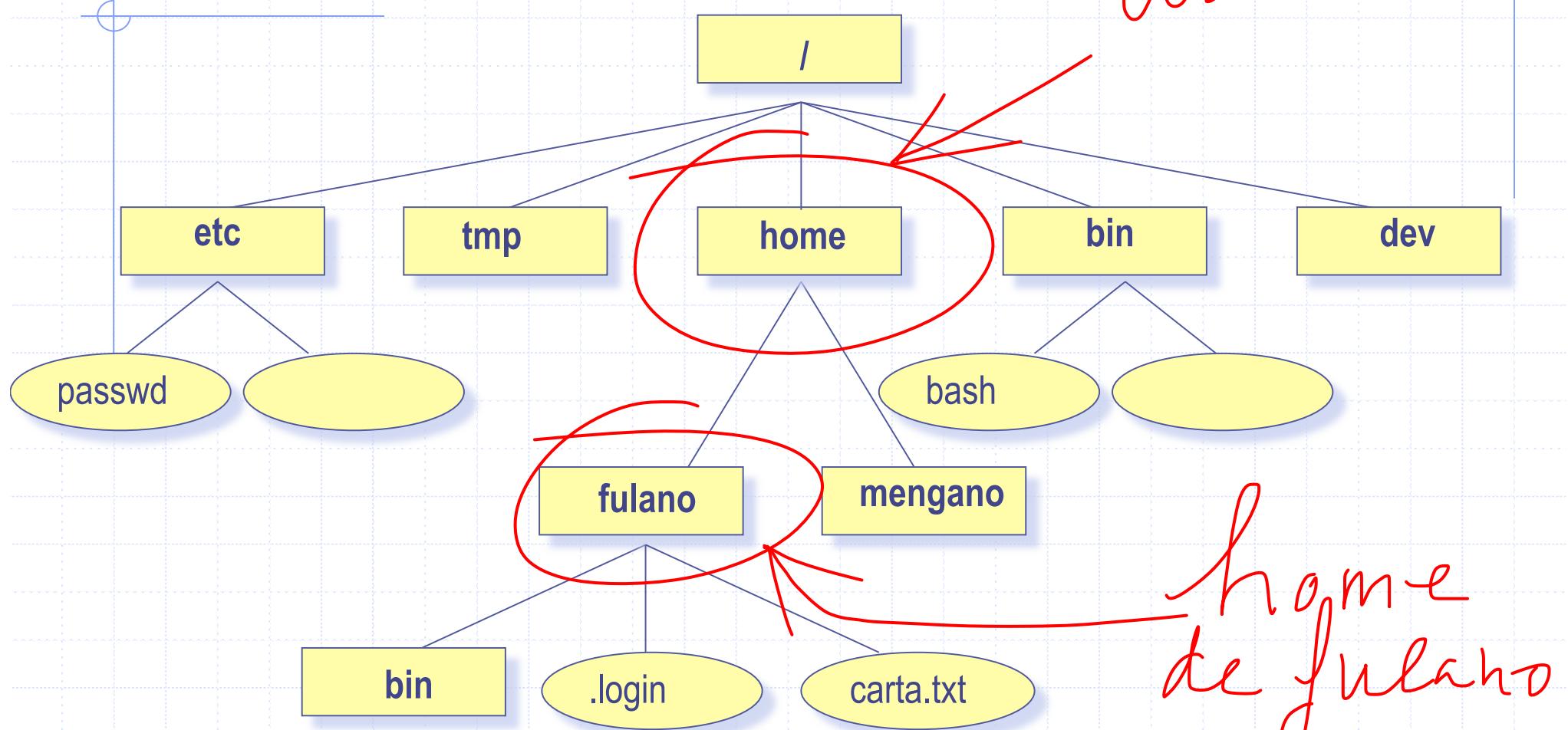
```
jcyelmo@Ubuntu-vb: ~/MOOC
jcyelmo@Ubuntu-vb:~/MOOC$ ls
page.html
jcyelmo@Ubuntu-vb:~/MOOC$ file page.html
page.html: HTML document, UTF-8 Unicode text
jcyelmo@Ubuntu-vb:~/MOOC$ cat page.html
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Título del documento</title>
</head>

<body>
Mi primera página
</body>
</html>

jcyelmo@Ubuntu-vb:~/MOOC$ cp page.html pagina.html
jcyelmo@Ubuntu-vb:~/MOOC$ ls
page.html pagina.html
jcyelmo@Ubuntu-vb:~/MOOC$ rm page.html
jcyelmo@Ubuntu-vb:~/MOOC$ ls
pagina.html
jcyelmo@Ubuntu-vb:~/MOOC$
```

Árbol de directorios

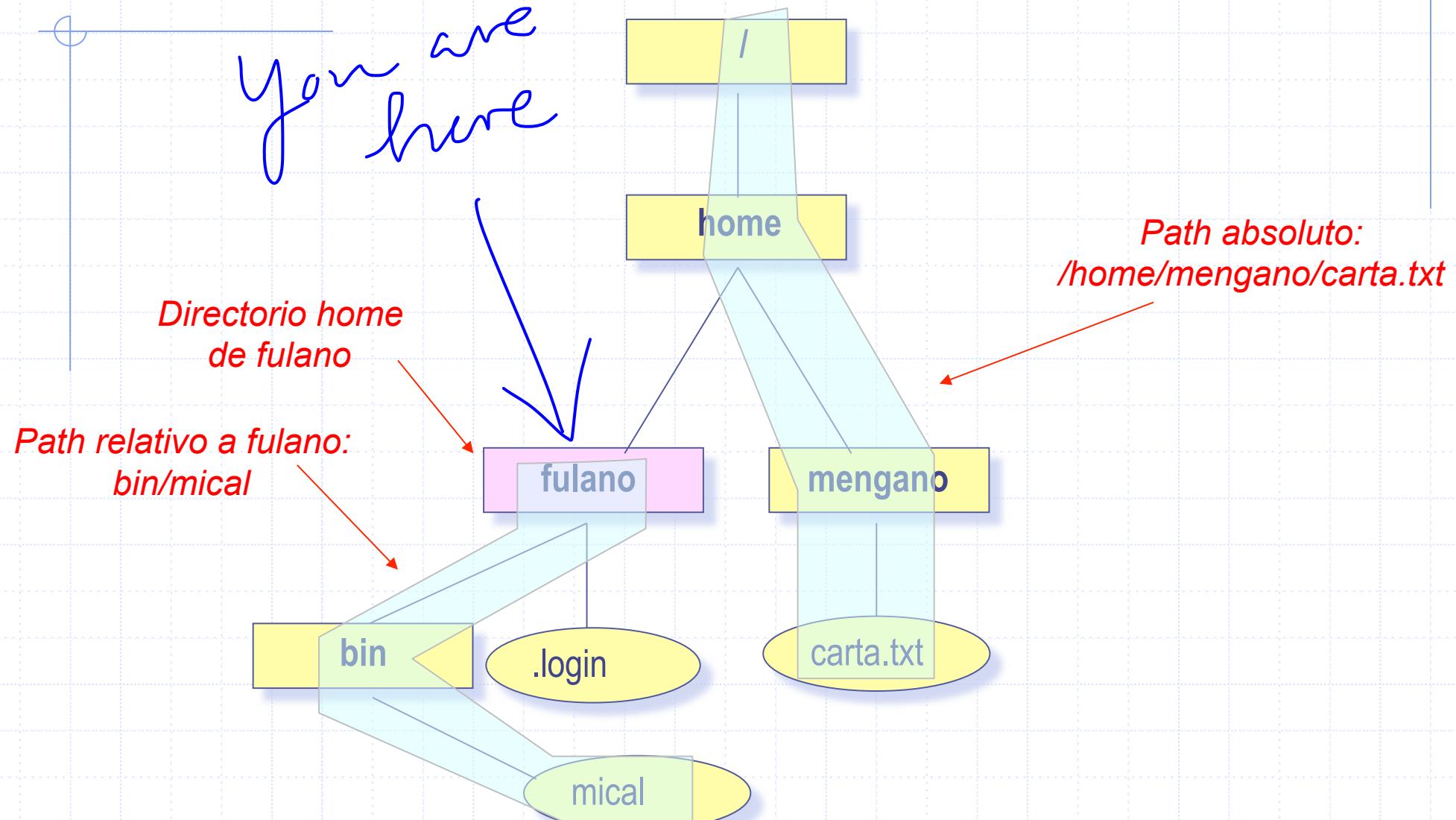
usuarios



Directarios en UNIX

- ◆ Los directorios son nodos del sistema de archivos y contienen otros nodos del sistema (ficheros o directorios)
- ◆ Los directorios son ficheros que contienen información sobre cómo encontrar otros ficheros
- ◆ El directorio inicial del sistema de archivos se denomina raíz (/)
- ◆ Todo directorio contiene al menos dos subdirectorios:
 - él mismo (.) y su antecesor (..)

Path de un fichero



Comandos básicos

Situarse y moverse por el sistema

`cd`

Cambio de directorio

`pwd`

¿Dónde estoy?

Crear y borrar directorios

`mkdir`

Crea directorios

`rmdir`

Borra directorios

Ocupación del sistema de archivos

`df`

Muestra el espacio disponible

`du`

Espacio ocupado por un subárbol
del sistema de archivos

Cambio de directorio

cd [directorio]

Cambia al directorio especificado o, en su defecto, al directorio home del usuario. Es un comando interno del intérprete de comandos (shell).

Ejemplos

cd

Cambia al directorio home del usuario

cd ~

Cambia al directorio home del usuario

cd ~fulano

Cambia al directorio home del usuario fulano

cd /home/fulano

Cambia al directorio home del usuario fulano

cd ..

Cambia al directorio superior

cd mibin

Cambia al subdirectorío “mibin” del directorio actual

Directorio en UNIX

```
jcyelmo@Ubuntu-vb: ~
jcyelmo@Ubuntu-vb:~/MOOC$ pwd
/home/jcyelmo/MOOC
jcyelmo@Ubuntu-vb:~/MOOC$ mkdir DirPrueba
jcyelmo@Ubuntu-vb:~/MOOC$ ls
DirPrueba pagina.html
jcyelmo@Ubuntu-vb:~/MOOC$ cd DirPrueba/
jcyelmo@Ubuntu-vb:~/MOOC/DirPrueba$ ls
jcyelmo@Ubuntu-vb:~/MOOC/DirPrueba$ cd ..
jcyelmo@Ubuntu-vb:~/MOOC$ rmdir DirPrueba/
jcyelmo@Ubuntu-vb:~/MOOC$ ls
pagina.html
jcyelmo@Ubuntu-vb:~/MOOC$ cd
jcyelmo@Ubuntu-vb:~$ pwd
/home/jcyelmo
jcyelmo@Ubuntu-vb:~$
```

Directorio de interés

/bin, /usr/bin	Comandos básicos del sistema: ls, mv, pwd, etc.
/usr/local	Comandos propios de la instalación local
/etc	Administración del sistema
/dev	Dispositivos: discos, red, impresoras, etc
/home	Usuarios del sistema
/tmp, /usr/tmp	Ficheros temporales con permiso para todos
/lib, /usr/lib	Bibliotecas del sistema



El sistema operativo UNIX

El sistema de archivos I

Fin del tema

Juan Carlos Yelmo



El sistema operativo UNIX

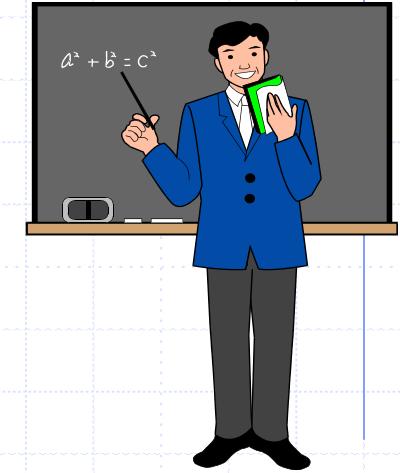
El sistema de archivos II

Juan Carlos Yelmo

Contenidos

2. El sistema de archivos

- Introducción
- Ficheros en UNIX
- Estructura del sistema de archivos
- Edición de ficheros



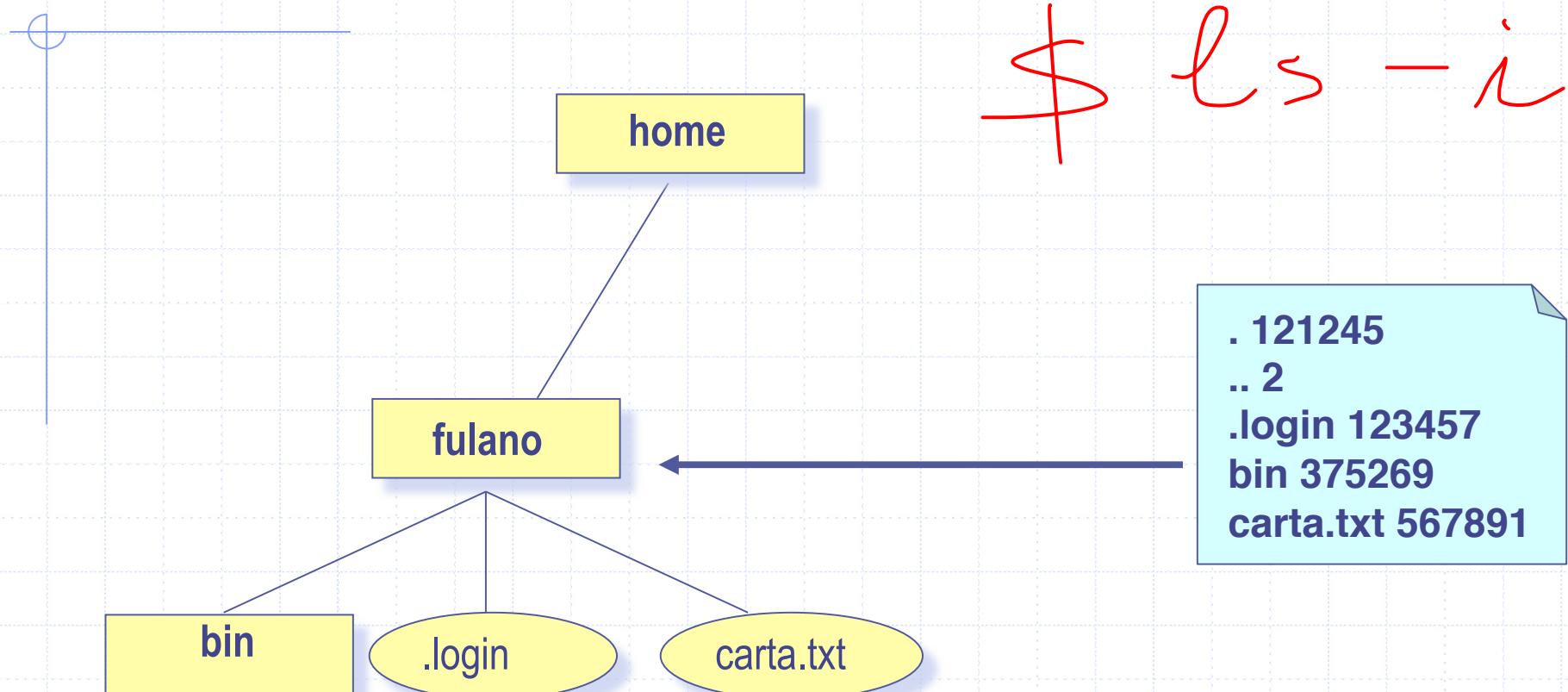
Estructura del sistema de archivos

- ◆ El sistema de archivos de UNIX se estructura como un árbol de nodos:
ficheros y directorios
- ◆ Un directorio es un fichero que contiene una lista de nodos, incluyendo una referencia a si mismo y a su ancestro
- ◆ UNIX guarda el sistema de archivos en disco como una lista de nodos: nodos-i

Los nodos índice (nodo-i)

- ◆ Representación interna de un fichero en UNIX.
- ◆ Contienen información de localización en disco del contenido del fichero e información adicional para la gestión y manipulación del fichero en el sistema de archivos
- ◆ Una entrada en un directorio (fichero) consta del nombre del fichero y el número de su nodo-i

Contenido de un directorio



Información en un nodo-i

- ◆ Modo.

- Tipo de fichero, modo de ejecución y permisos de acceso

- ◆ Número de enlaces al fichero

- ◆ Identificación de propietario y grupo

- ◆ Tamaño del fichero en bytes

- ◆ Fecha y hora de último acceso, modificación y cambio

Información en un nodo-i

- ◆ Dispositivo donde está almacenado el fichero
- ◆ Dirección de los bloques de disco que componen el fichero
- ◆ Tamaño óptimo del bloque de disco
- ◆ Número de bloques de disco asignados al fichero

Nodos-i

Lista de nodos-i



Un nodo-i

Modo	Núm. enlaces	UID	GID	Tamaño	Fechas	Bloques en disco
------	-----------------	-----	-----	--------	--------	------------------

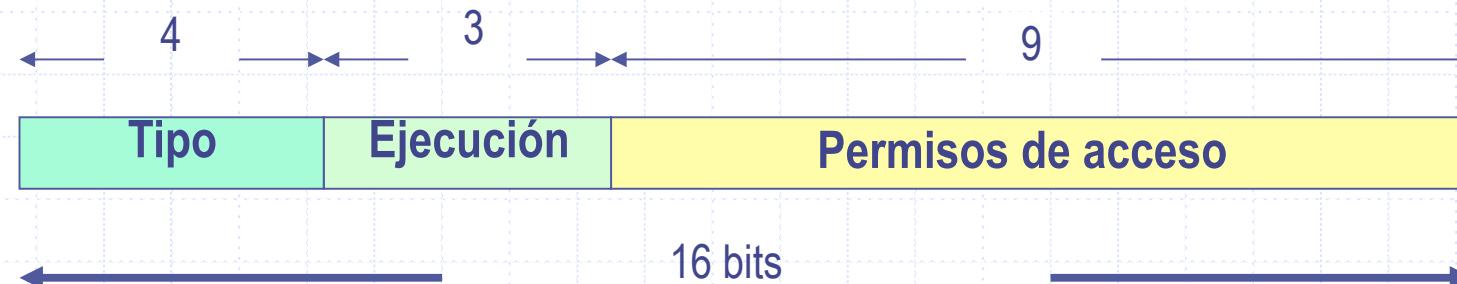
Tipo	Modo ejec.	Permisos
------	---------------	----------

Acceso	Modif.	Cambio
--------	--------	--------

r/w Contenido metadatos

Modo de ficheros

- ◆ Entero de 16 bits que codifica el tipo de fichero, forma de ejecución y permisos de acceso



Tipo de fichero

- ◆ Se establece cuando se crea el fichero y no se puede cambiar
 - Fichero normal (-)
 - Directorio (d)
 - Fichero pipe (p)
 - Enlace simbólico (l)
 - Dispositivo de almacenamiento por caracteres (c)
 - Dispositivo de almacenamiento por bloques (b)
 - ...

Permisos de acceso

- ◆ Cada fichero en UNIX tiene un propietario y un conjunto de *permisos* asociados que determinan qué puede hacerse con él y quién puede hacerlo
- ◆ Hay tres tipos de permisos por fichero: lectura (r), escritura (w) y ejecución (x)
- ◆ Estos permisos se pueden asignar a tres tipos de usuario: el propietario (*user*), los miembros del grupo del propietario (*group*) y terceras personas (*others*)

Permisos de acceso

Tipo	Ejecución	Permisos de acceso
------	-----------	--------------------

Usuario	Grupo	Otros
---------	-------	-------

1: s10: n

r w x

lectura

ejecución

escritura

Permisos de acceso

- ◆ Los permisos de acceso de un fichero pueden cambiarse `chmod`
- ◆ Existe un tipo de usuario especial (*el superusuario*) que puede leer, modificar o ejecutar cualquier fichero del sistema
- ◆ El identificador de acceso *root* (el administrador del sistema) tiene permisos de superusuario

Cambiar permisos de acceso

chmod permisos fichero

Cambia los permisos de un fichero a los valores proporcionados

Ejemplos

chmod a+r page.html	Añade permiso de lectura del fichero <i>page.html</i> para todos (usuario, grupo y otros)
chmod g+w page.html	Añade permiso de escritura del fichero <i>page.html</i> para los miembros del grupo del propietario
chmod u-x page.html	Elimina el permiso de ejecución del fichero <i>page.html</i> para el propietario
chmod +x comando	Añade permiso de ejecución del fichero <i>comando</i> para todos (usuario, grupo y otros)
chmod 700 comando	Añade permisos de lectura, escritura y ejecución del fichero <i>comando</i> para el usuario y elimina todos los permisos para los miembros del grupo y otros (permisos en octal)

Listar ficheros

ls [opciones] [ficheros]

Lista los ficheros contenidos en el directorio actual o los nombrados explícitamente como argumentos

Opciones principales

-l	Listado en formato largo. Incluye permisos, propietario, tamaño, última modificación, etc.
-t	Listar por orden de fecha/hora de última modificación, primero el más reciente
-r	Listar en orden inverso. Para combinar con otras opciones
-a	Listado que incluye ficheros ocultos (e.g. .login)
-i	Listado que incluye el número de nodo-i

Ejemplos

ls -lt *.html	Lista en formato largo y por orden de antigüedad los ficheros cuyo nombre acaba en .html
----------------------	--

Metacaracteres

- ◆ Caracteres especiales utilizados para nombrar grupos de ficheros de forma simbólica
 - *: Cualquier cadena de caracteres
 - ◆ rm *.html: Borra los ficheros que acaban en .html
 - ?: Cualquier carácter individual
 - ◆ ls modulo.?: Lista modulo.c, modulo.o, etc.
 - [c1,c2,...,cn] o [c1-cn]: Cualquier carácter dentro de una enumeración o rango
 - ◆ ls capitulo[1-9]: Lista capitulo1, ..., capitulo9

Sesión con nodos-i

Ubuntu 14.04 [Running]

Terminal

```
jcyelmo@Ubuntu-vb: ~/MOOC
jcyelmo@Ubuntu-vb:~/MOOC$ ls
carta.txt page.html pagina.html
jcyelmo@Ubuntu-vb:~/MOOC$ ls -i
182855 carta.txt 182874 page.html 182792 pagina.html
jcyelmo@Ubuntu-vb:~/MOOC$ ls -ai
182808 . 182512 .. 182855 carta.txt 182874 page.html 182792 pagina.html
jcyelmo@Ubuntu-vb:~/MOOC$ ls -li page.html
182874 -rw-rw-r-- 1 jcyelmo jcyelmo 142 ene 17 14:04 page.html
jcyelmo@Ubuntu-vb:~/MOOC$ stat page.html
  File: 'page.html'
  Size: 142          Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 182874      Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/ jcyelmo)  Gid: ( 1000/ jcyelmo)
Access: 2015-01-17 14:04:27.624749751 +0100
Modify: 2015-01-17 14:04:27.624749751 +0100
Change: 2015-01-17 14:04:27.624749751 +0100
 Birth: -
jcyelmo@Ubuntu-vb:~/MOOC$ ls *.html
page.html pagina.html
jcyelmo@Ubuntu-vb:~/MOOC$
```



El sistema operativo UNIX

El sistema de archivos II

Fin del tema

Juan Carlos Yelmo



El sistema operativo UNIX

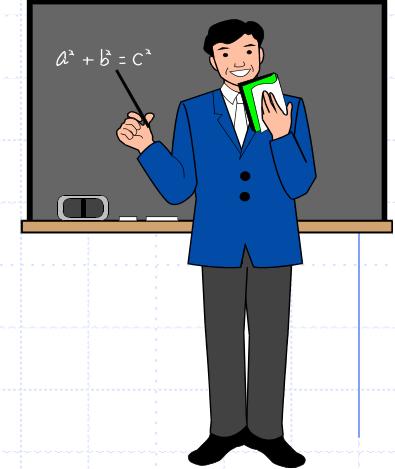
El sistema de archivos III

Juan Carlos Yelmo

Contenidos

2. El sistema de archivos

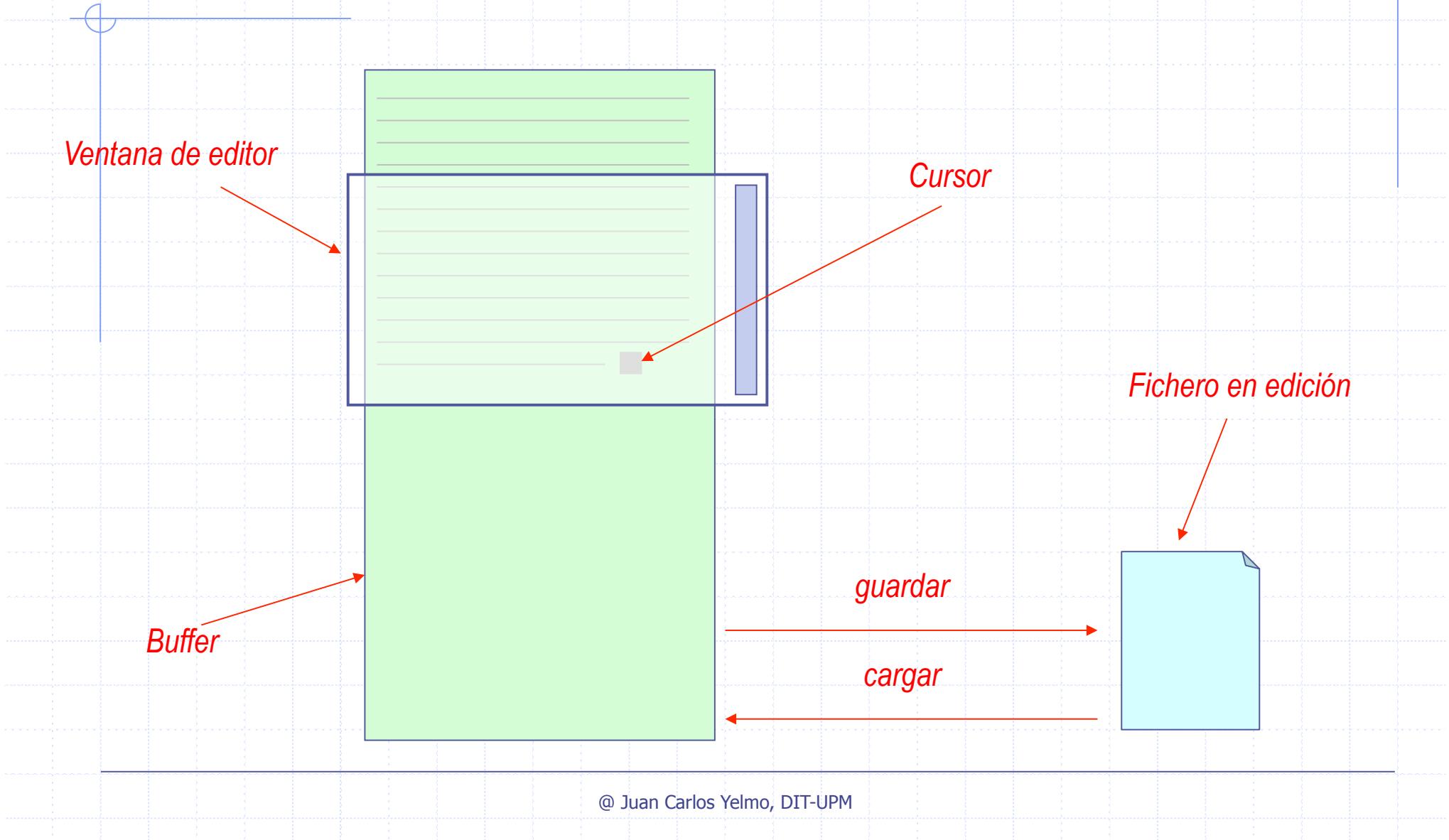
- Introducción
- Ficheros en UNIX
- Estructura del sistema de archivos
- Edición de ficheros



Edición de ficheros

- ◆ Un editor es un programa que se utiliza para crear o modificar ficheros que contienen texto simple: código fuente, datos de configuración, html, etc.
- ◆ El editor crea un buffer temporal donde almacena el fichero editado
- ◆ La pantalla del terminal actúa de ventana a través de la que se visualiza parte del buffer. Esta ventana puede deslizar arriba y abajo en el buffer
- ◆ Ejemplos de editores de texto: Notepad (Windows),TextEdit (OS X), Sublime Text, Emacs, Vi, Vim....

Editor de pantalla



Editores de texto

```
jcyelmo@Ubuntu-vb: ~/MOOC
!doctype html
<html>
<head>
<meta charset="utf-8">
<title>Mi primera página</title>

<style type="text/css">
body {
    background-color: #FF1;
}
p {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 24px;
    color: red;
}
</style>
</head>

<body>
<p> Mi primera página Web con estilo</p>
</body>

</html>
```

Vi

```
primeraCSS.html  UNREGISTERED
primeraCSS.html
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>Mi primera página</title>
6
7  <style type="text/css">
8  body {
9      background-color: #FF1;
10 }
11 p {
12     font-family: Arial, Helvetica, sans-serif;
13     font-size: 24px;
14     color: red;
15 }
16 </style>
17 </head>
18
19 <body>
20 <p> Mi primera página Web con estilo</p>
21 </body>
22
23 </html>
```

23 lines, 308 characters selected Tab Size: 4 HTML

Sublime Text →

El editor vi

- ◆ Desarrollado para una de las primeras versiones de BSD UNIX
- ◆ Estándar de facto en UNIX y omnipresente en todas sus variantes
- ◆ Muy utilizado por su potencia, difusión, necesitar pocos recursos, velocidad de arranque y por ser editor secundario de otras aplicaciones
- ◆ Tanto el texto como los comandos propios del editor se introducen desde el teclado
- ◆ Puede resultar incómodo de utilizar al principio

El editor vi

vi [opciones] [ficheros]

Editor de pantalla

Opciones principales

-r file

Recupera y edita “file” después de una caída del sistema o del editor vi

+n

Edita el fichero en la línea n

Ejemplos

vi file

Edita el fichero file

vi +20 geo.htm

Edita el fichero geo.htm y sitúa el cursor en la línea 20

vi file1 file2

Carga file1 y file2 y edita file1. file2 se edita con el comando :n

El editor vi

- ◆ El editor vi tiene dos modos
 - **Modo comando.** Lo que el usuario teclea se interpreta como un comando de vi
 - **Modo inserción.** Lo que el usuario teclea se interpreta como texto a insertar
- ◆ Inicialmente, vi arranca en modo comando
 - Para pasar a modo inserción: i
 - Para volver a modo comando: ESC

Vi. comandos básicos

Pasar a modo inserción

i	Antes del cursor
a	Detrás del cursor
o	Al comienzo de nueva línea

Salir de vi

zz	Guardar y salir
:q	Salir sin guardar (pregunta)
^z	Suspender edición (se recupera con fg)

Otros

dd	Borrar línea donde está el cursor
u	Deshacer último cambio
/<texto>	Buscar <texto> hacia adelante
:r fichero	Inserta el contenido de fichero

ctrl-z



El sistema operativo UNIX

El sistema de archivos III

Fin del tema

Juan Carlos Yelmo



El sistema operativo UNIX

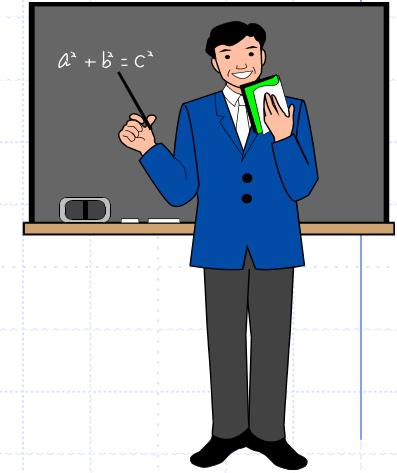
La interfaz de usuario I

Juan Carlos Yelmo

Contenidos

3. La interfaz de usuario

- Las shells de UNIX
- Sintaxis y entorno de comandos
- Redirección de entrada y salida
- Composición de comandos
- Variables de la Shell
- Inicialización de sesión



Introducción

- ◆ Interfaz de usuario de un sistema software
 - Conjunto de aspectos de un sistema software que un usuario puede ver (en general percibir) y los comandos y mecanismos que el usuario utiliza para operar el sistema e introducir datos.
 - Hay dos tipos básicos: GUIs e interfaces textuales o de línea de comandos
- ◆ La shell es la interfaz de usuario del sistema operativo UNIX
- ◆ Es un intermediario entre el usuario y el kernel del sistema operativo

La shell

- ◆ UNIX puede tener varias interfaces, textuales o gráficas, sustituibles y configurables por el usuario
- ◆ La shell original es un intérprete de línea de comandos: Bourne Shell (sh)
- ◆ La shell estándar de LINUX es una reimplementación de la Bourne Shell: Bourne Again Shell (**bash**)
- ◆ Hay interfaces gráficas a UNIX similares a los entornos MS-Windows: OS X Aqua, GNOME (GNU), KDE, **Unity**, etc.

Usos de la Shell de UNIX

◆ Interacción con el usuario

◆ Personalización de sesión

◆ Programación por combinación de comandos (*shell script*)

| |
| bash
| |

Usos de la Shell

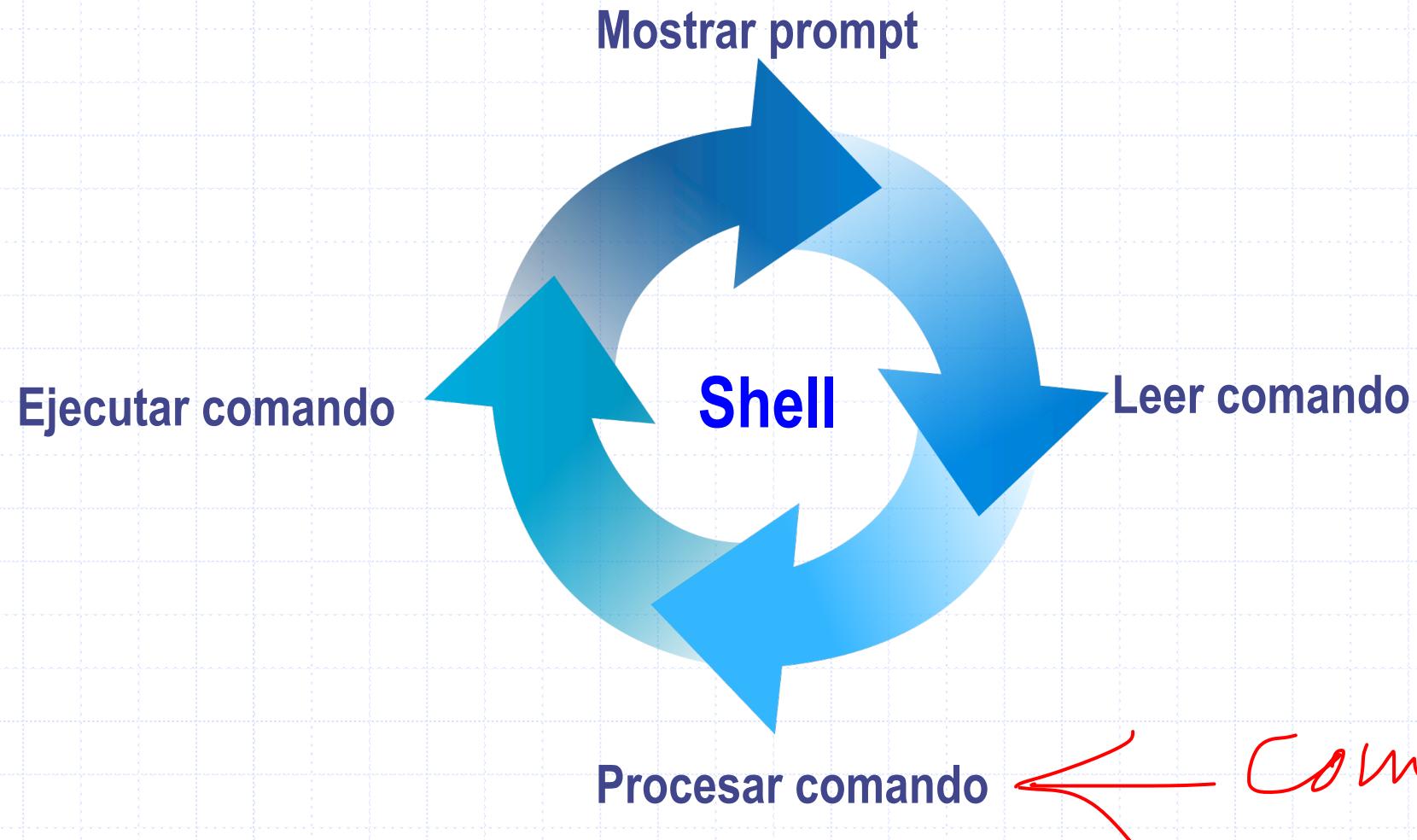
◆ Uso interactivo

- La shell muestra el prompt, espera que el usuario introduzca comandos y argumentos, analiza sintácticamente la línea, ejecuta el comando y devuelve el prompt
- Tiene facilidades de uso de metacaracteres, redireccionamiento de entrada-salida, combinación de comandos, historia de comandos, etc.



más adelante

Bucle de procesamiento de comandos



Usos de la Shell

más adelante

◆ Personalización de la sesión UNIX

- La Shell define variables que controlan la sesión de usuario. Definiendo estas variables el usuario puede configurar la sesión según gustos y necesidades
- La sesión de usuario se configura inicialmente a través de ficheros que se leen cuando se inicia la shell. En estos ficheros se pueden definir variables y comandos que se ejecutan al comienzo de la sesión

Usos de la Shell

◆ Programación

- La shell permite combinar comandos UNIX en un fichero (*shell script*) y proporciona comandos de control típicos de programación: bucle, bifurcación, etc. Esto convierte a la shell en un lenguaje de programación de alto nivel para escribir comandos nuevos o combinaciones frecuentes en base a comandos preexistentes

Los comandos UNIX

- ◆ Un comando (orden) es una cadena de caracteres que indica a la shell la realización de una acción específica
- ◆ Normalmente la acción es la búsqueda de un programa o shell script en el sistema de archivos y su ejecución o interpretación, aunque la shell tiene también comandos internos (*built in*)
- ◆ Un comando suele admitir opciones y argumentos

Los comandos de UNIX

◆ Sintaxis de comandos de la shell de UNIX

```
$ comando [ opciones ] [ argumentos ]
```

Programa/acción a ejecutar

Modificadores

Datos de entrada

Los comandos UNIX

◆ **Comando.** Casi siempre el nombre de un fichero en el sistema de archivos

- Con path absoluto: /bin/ls
- Búsqueda en variable PATH: ls

◆ **Opciones.** Modificadores de la acción básica realizada por el comando

- Casi siempre precedidas de “-”
- Casi siempre una letra: -a
- Casi siempre acumulativas:
 - ◆ ls -a -l = ls -al

◆ **Argumentos:** ficheros, máquinas, usuarios...

Argumentos: ficheros

¿Cuál?

which comando

Ejemplo:

```
$ which pwd
```

Devuelve el path absoluto del comando

Devuelve el path del comando pwd

¿Dónde está?

find path opciones

Ejemplo:

```
$ find . -name  
page.html -print
```

Encuentra ficheros a partir de *path*

Busca a partir del directorio actual la localización del fichero page.html

¿Qué ha cambiado?

diff file1 file2

Ejemplo:

```
$diff filev1 filev2
```

Muestra las diferencias entre dos ficheros de texto línea a línea

Muestra las diferencias entre dos versiones de un fichero

Argumentos: máquinas

Conexión segura

ssh hostname

Ejemplo:

```
$ ssh -T  
git@github.com
```

Login remoto seguro en la máquina especificada

Comprueba accesibilidad ssh a github (hay que autenticarse)

Alcanzabilidad

ping hostname

Ejemplo:

```
$ ping google.com
```

Comprueba si la máquina es alcanzable en red y mide tiempo de eco

Comprueba que google.com es alcanzable

Transferencia de archivos

ftp hostname

Ejemplo:

```
$ ftp  
ftp.mozilla.org
```

Establece conexión para transferencia de archivos

Establece conexión con el servidor público de Mozilla (anónimo)



El sistema operativo UNIX

La interfaz de usuario I

Fin del tema

Juan Carlos Yelmo



El sistema operativo UNIX

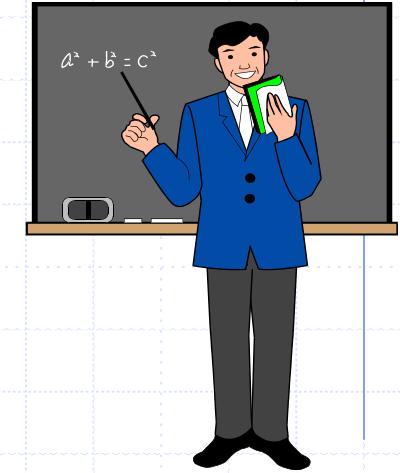
La interfaz de usuario II

Juan Carlos Yelmo

Contenidos

3. La interfaz de usuario

- Las shells de UNIX
- Sintaxis y entorno de comandos
- Redirección de entrada y salida
- Composición de comandos
- Variables de la Shell
- Inicialización de sesión

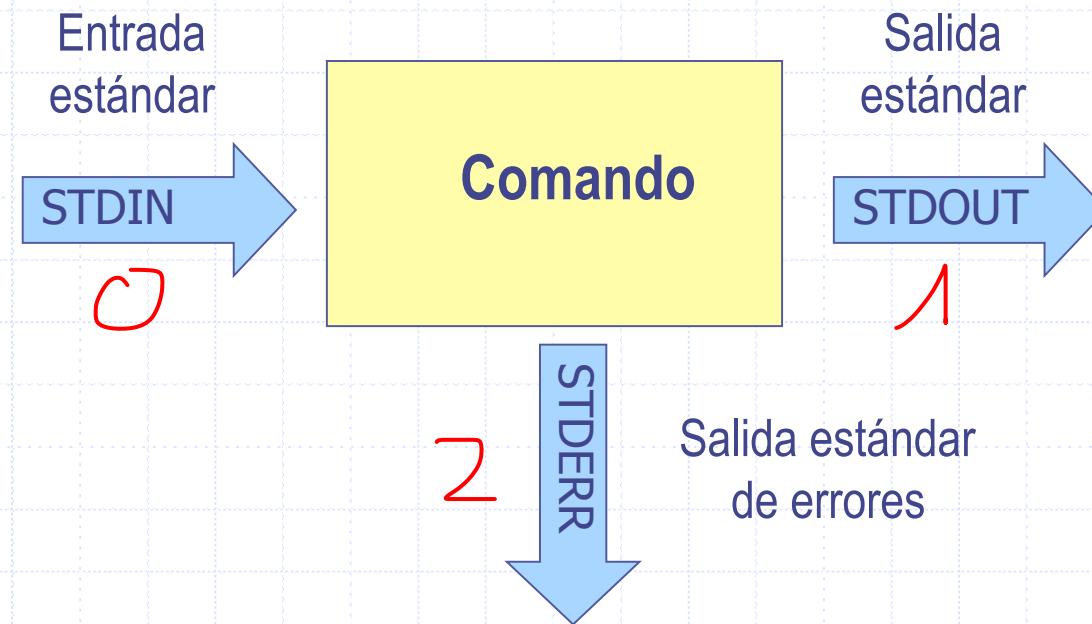


Entrada/salida de comandos

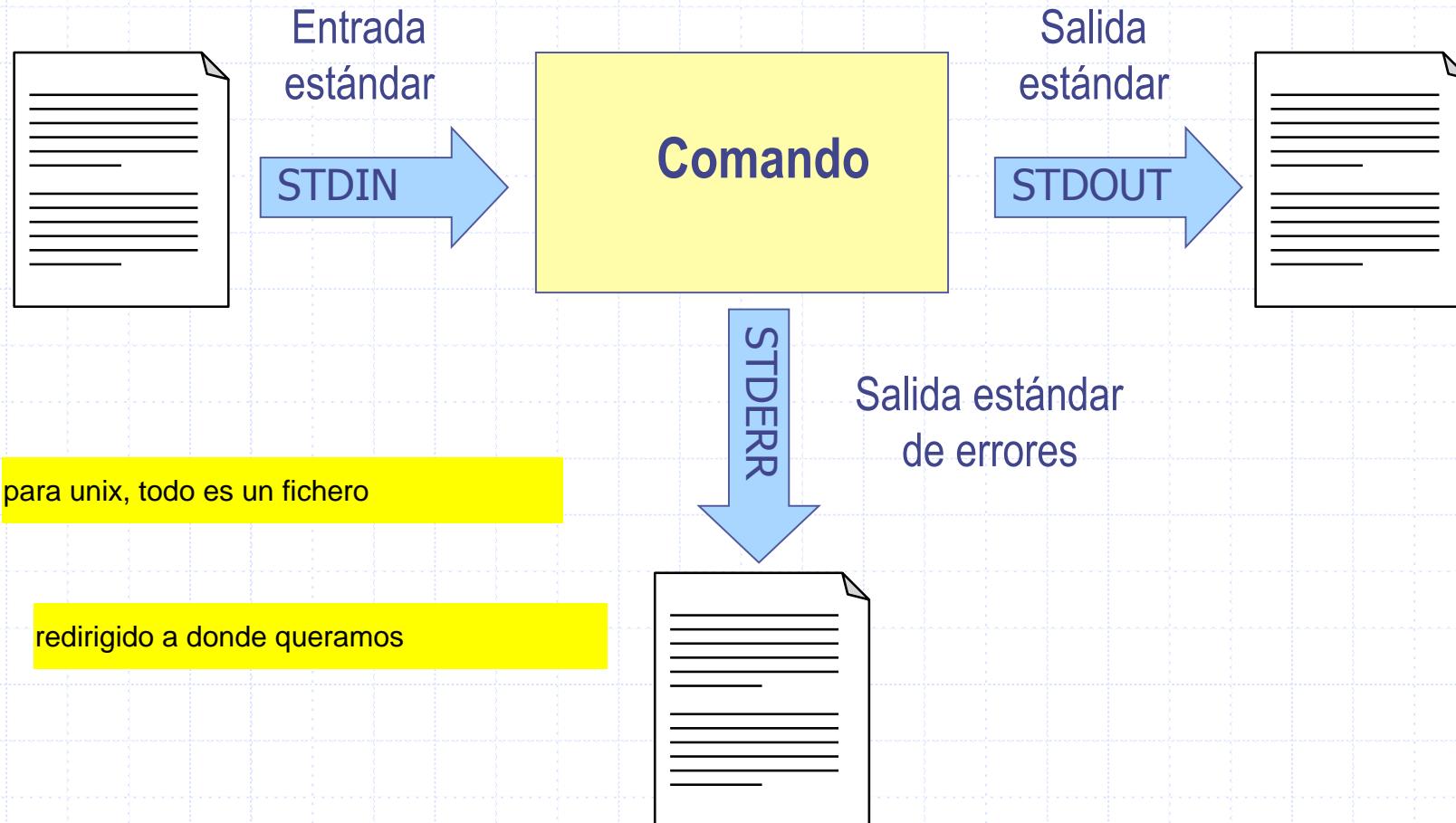
- ◆ Los argumentos de un comando suelen indicar la fuente de información de entrada (o el destino de los resultados de salida)
- ◆ Además de los argumentos, un comando UNIX típico tiene definidos unos canales (ficheros) de entrada y salida por defecto:
 - Entrada estándar (por defecto el teclado)
 - Salida estándar (por defecto la pantalla)
 - Salida estándar de errores (por defecto la pantalla)

terminal

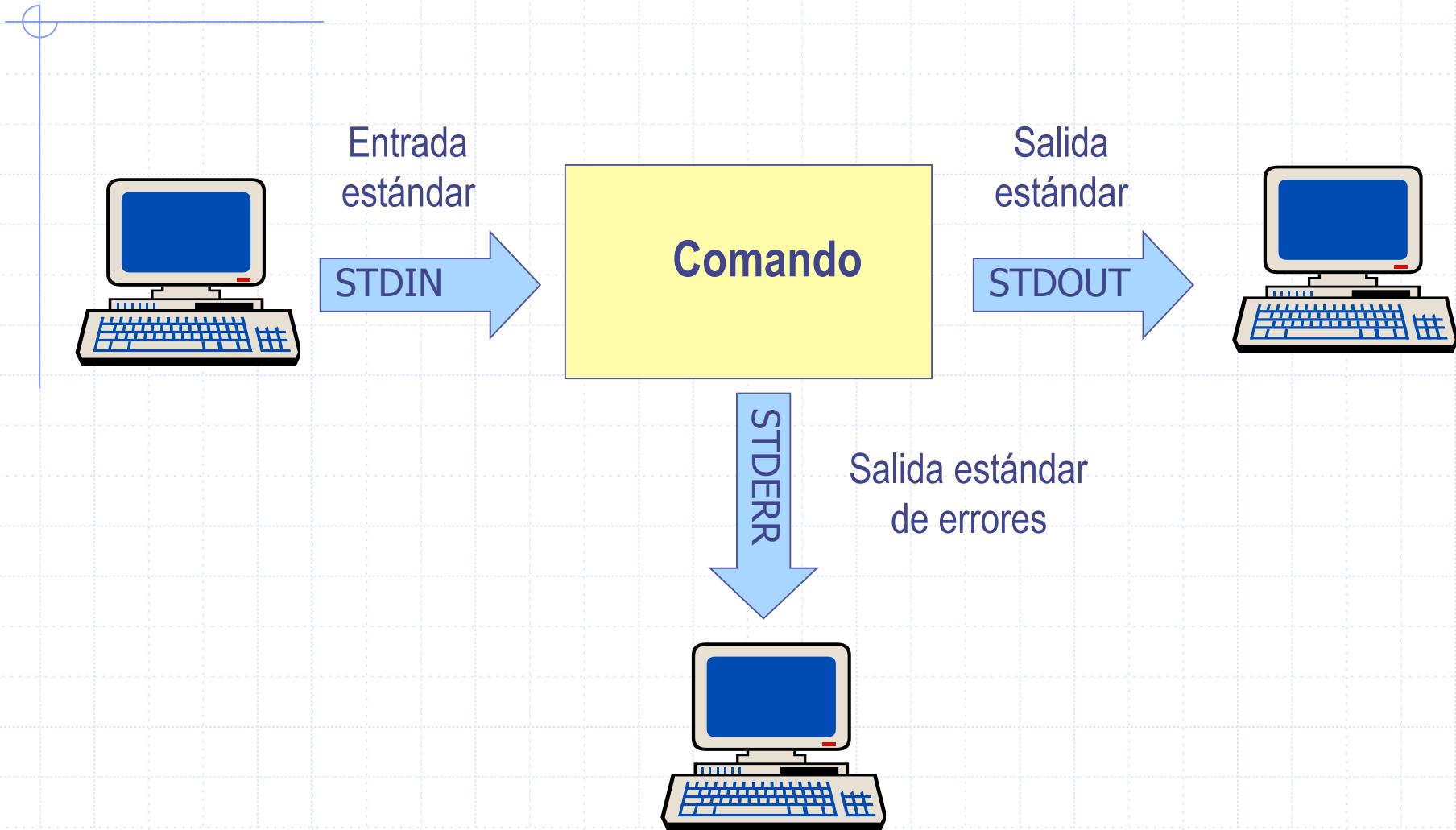
Entrada/salida de comandos



Entrada/salida de comandos



Entrada/salida de comandos



Entradas de un comando

- ◆ Sus opciones y argumentos.

- Son la entrada básica y explícita de información para el comando

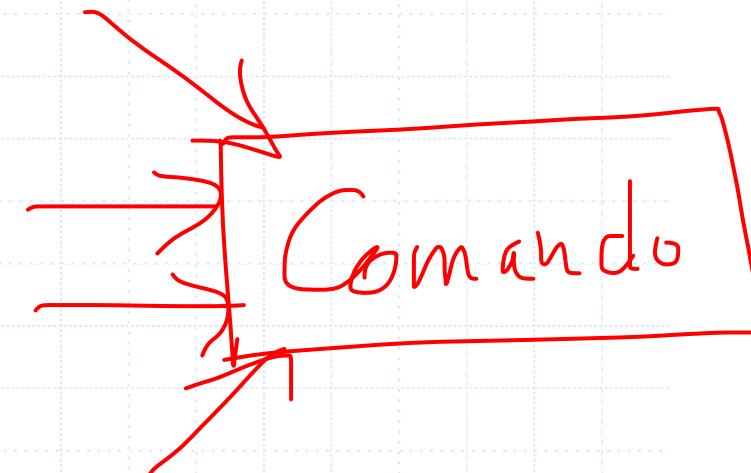
- ◆ Ficheros predefinidos

- Ficheros de personalización de sesión (.login) o de configuración propios del comando (.exrc para vi)

- ◆ Variables de entorno

- TERM, PATH, PAGER, etc.

- ◆ La entrada estándar



Salidas de un comando

◆ Ficheros

- Nombre dado como argumento del comando

\$ gcc programa.c -o programa *programa*

- Nombre por defecto

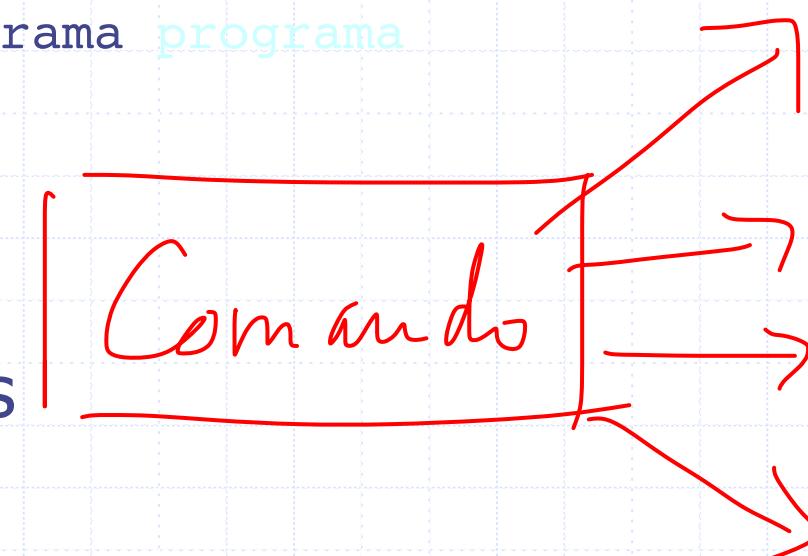
\$ gcc programa.c (a.out)

◆ Salida estándar

◆ Salida estándar de errores

◆ Valor de retorno a la shell

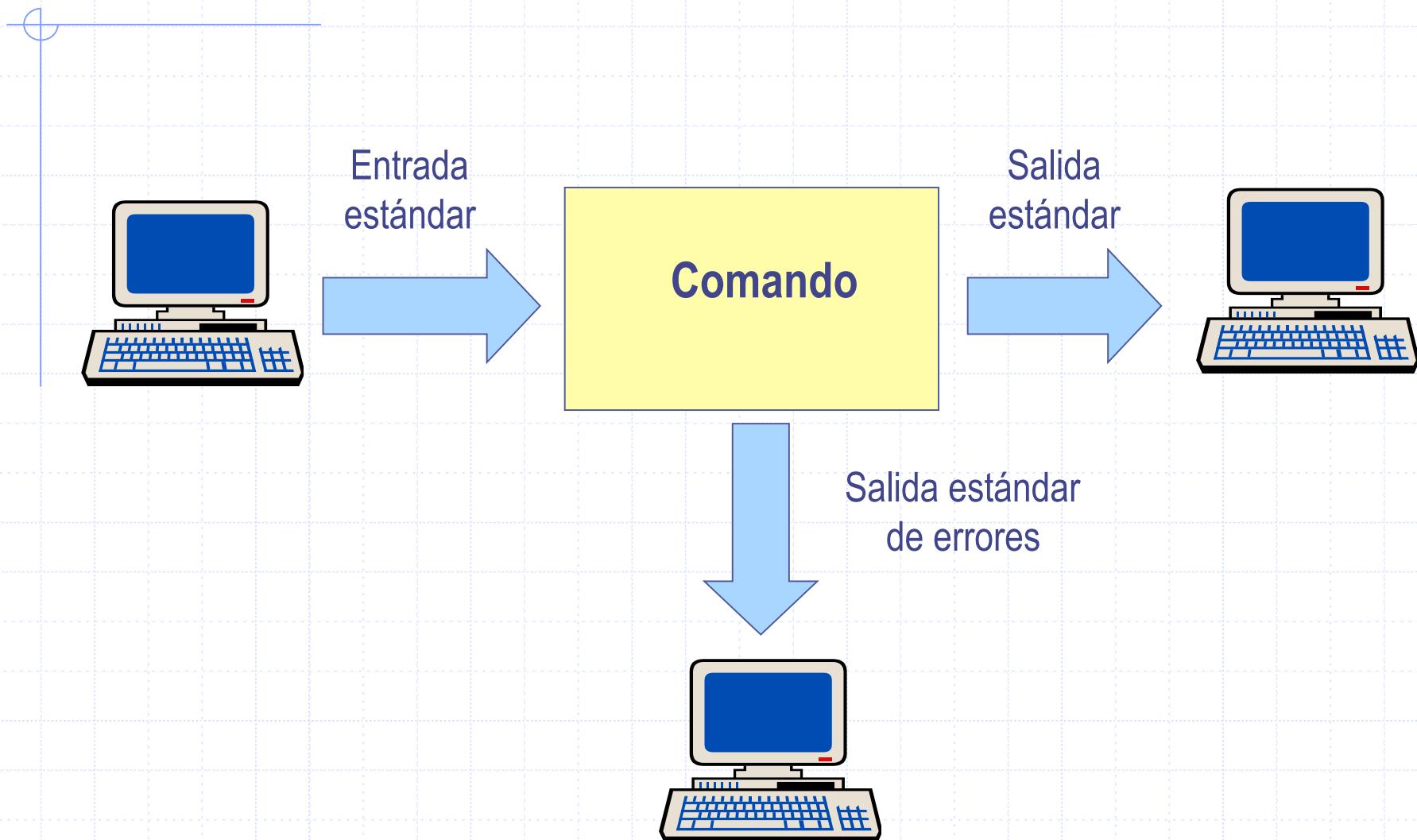
- Cero (0) es ejecución con éxito, cualquier otro entero es ejecución con errores



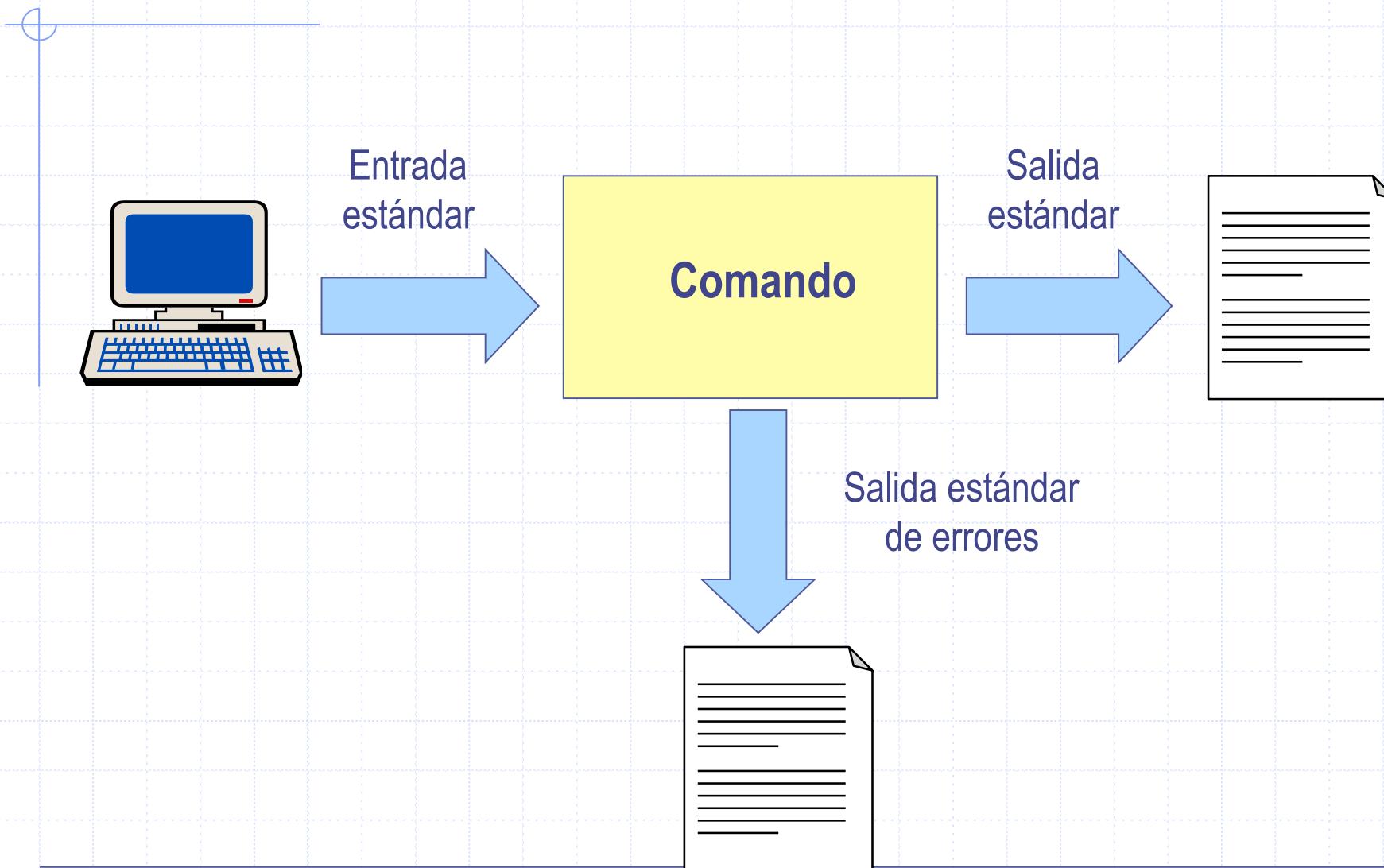
Redirección de entrada/salida

- ◆ Los canales de entrada y salida estándar de un comando se asocian por defecto al terminal de usuario, pero pueden asociarse a cualquier otro fichero (redirección de entrada/salida)
- ◆ La salida estándar de un comando puede asociarse con la entrada estándar de otro para conseguir la composición de comandos en cadena (pipeline)

Redirección de salida



Redirección de salida



Redirección de salida estándar

◆ Redirigir salida estándar

- comando > fichero
- \$ cal > calendario

◆ Añadir salida a fichero preexistente

- comando >> fichero
- \$ cal >> calendario

◆ Redirigir salida estándar a salida de error

- comando >&2
- \$ echo "Error" >&2

Redirección de entrada estándar

◆ Redirigir entrada estándar

- comando < fichero
- \$ wc < poema

◆ Entrada en la misma línea de comando (here document)

- comando << marca

...

marca

- \$ wc << fin

Volverán del amor en tus oídos
las palabras ardientes a sonar;

fin

ctrl-d para terminar

Redirección. Resumen

Salida estándar (stdout)

Defecto	> file	>> file	>&2
Pantalla	Redirigir a fichero	Añadir a fichero	Combinar con stderr
	>! machaca file si existe	>>! crea fichero si no existe	

Redirección. Resumen

Entrada estándar (stdin)

Defecto	< file	<< END
Teclado	Entrada desde fichero	Entrada en la misma línea de comando hasta la marca END
		Here document La marca de fin de entrada puede ser cualquiera

Composición de comandos

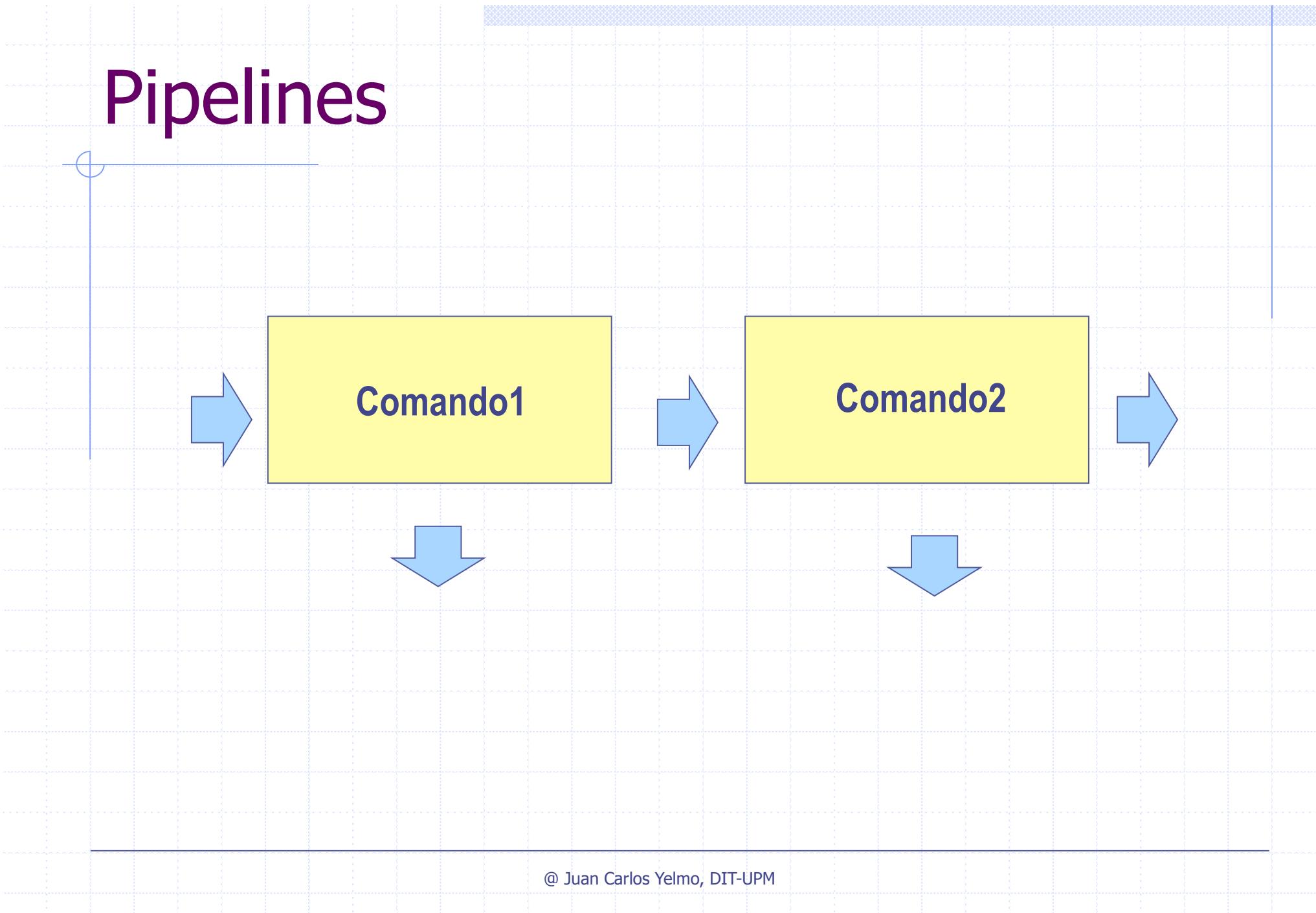
◆ Secuencia de comandos

- comando1; comando2
- \$ date; who

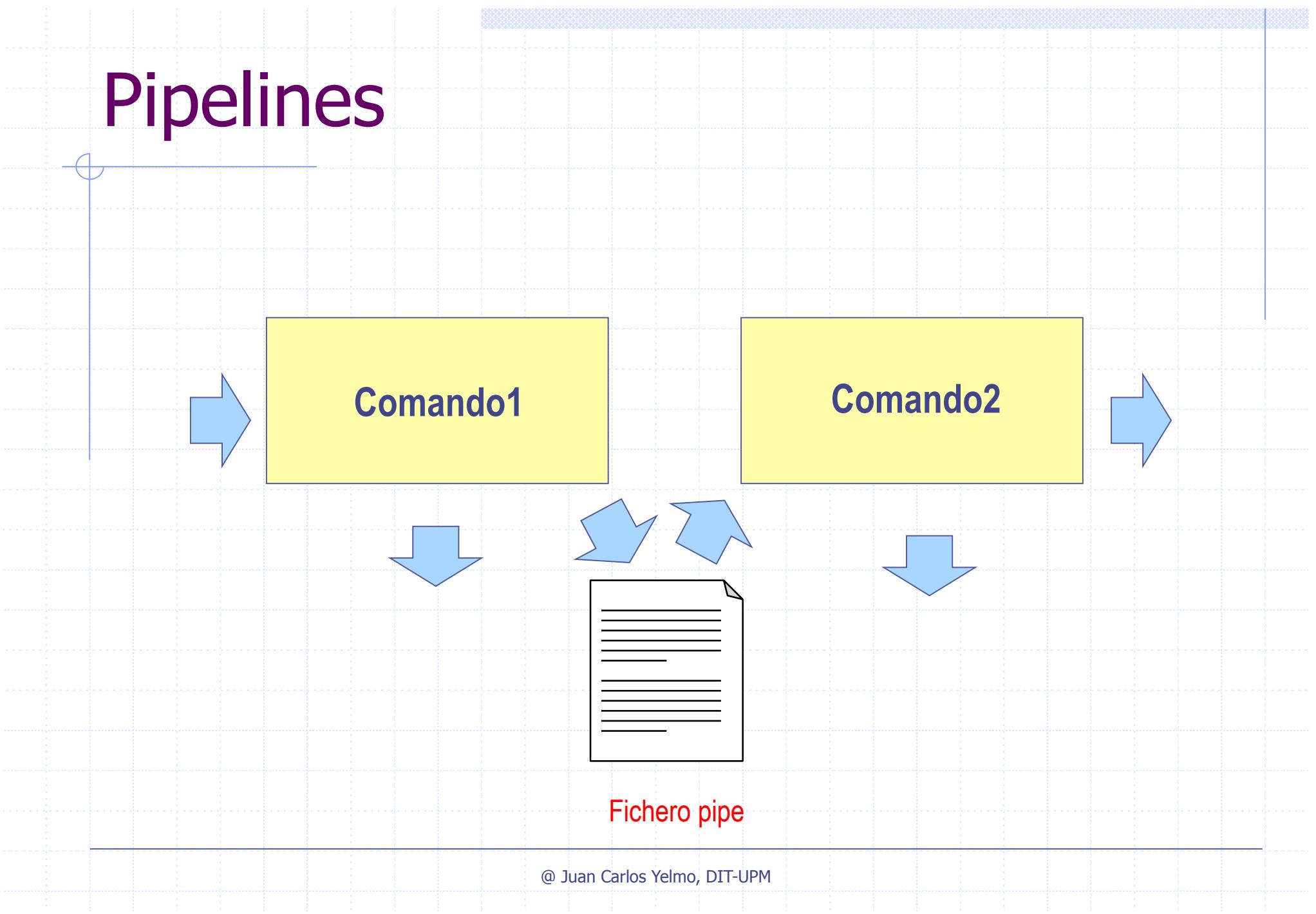
◆ Encadenamiento de comandos con un solo flujo de información (pipeline)

- comando1 | comando2 ...
- \$ grep -i amor poema | wc -l

Pipelines



Pipelines



Comandos UNIX

Cuenta palabras

wc opciones fichero

Ejemplo:

```
$ wc -l soneto.txt
```

Cuenta caracteres, líneas y palabras en *fichero*. Sin *fichero* usa la entrada estándar

Cuenta líneas en el fichero *soneto.txt*

Búsqueda de patrones

grep opciones patron file

Ejemplo:

```
$ grep -w cara  
fichero.txt
```

Busca palabras que encajan con el patrón. El patrón puede representar un conjunto de cadenas de caracteres mediante metacaracteres y otras expresiones

Busca la palabra *cara*, completa, en el fichero *fichero.txt* e imprime las líneas que la contienen. No mostraría, por ejemplo, líneas que contengan la palabra *caramelo*



El sistema operativo UNIX

La interfaz de usuario II

Fin del tema

Juan Carlos Yelmo



El sistema operativo UNIX

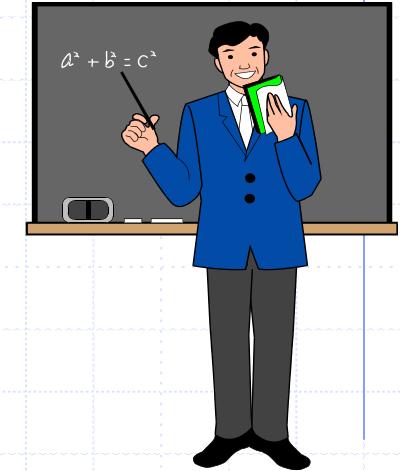
La interfaz de usuario III

Juan Carlos Yelmo

Contenidos

3. La interfaz de usuario

- Las shells de UNIX
- Sintaxis y entorno de comandos
- Redirección de entrada y salida
- Composición de comandos
- Variables de la Shell
- Inicialización de sesión



Variables de shell

- ◆ La shell permite la asignación y consulta de valores a variables. Estas variables pueden ser utilizadas por la propia shell o por los comandos y programas ejecutados por ésta
- ◆ Los valores de las variables son cadenas de cero o más caracteres
- ◆ Un identificador de variable puede ser cualquier palabra que comience con una letra y continúe con letras, dígitos y el carácter de subrayado

Uso de variables

◆ Asignación

- En sh y bash

- ◆ \$ nombre=manu

sin espacios

◆ Uso

- ◆ \$ echo \$nombre

manu

- ◆ \$ echo "\${nombre} perez"

manu perez

- ◆ \$ echo "\${nombre}el perez"

manuel perez

Tipos de variables

- ◆ **Variables ordinarias.** Variables locales de propósito general
- ◆ **Variables de entorno.** Describen el contexto de ejecución y se heredan
- ◆ **Variable especiales de la shell.**
Configuran el entorno de la propia shell.
 - Variables para parámetros posicionales en la invocación de comandos

Variables de entorno

◆ Describen el entorno de ejecución:

- TERM: Tipo de terminal
- PATH: Lista de directorios por defecto para búsqueda de comandos
- HOSTNAME: Nombre de la máquina
- USER: Nombre del usuario
- SHELL: Shell por defecto
- HOME: Directorio base del usuario
- etc.

Variables de entorno

◆ Asignación (sh y bash)

- \$ TERM=vt100; export TERM
- \$ PATH=\$PATH:/home/jcyelmo; export PATH
- \$ export PATH=\$PATH:/home/jcyelmo

◆ Consulta (sh y bash)

- \$ echo \$PATH
- \$ printenv

Variables posicionales

◆ Albergan los parámetros de entrada a los comandos

- \$ <comando> <arg1> arg2> ...
- \$0 \$1 \$2 ...
- \$ cc -o programa parte1.c parte2.c
 - ◆ \$0= cc
 - ◆ \$1= -o
 - ◆ \$2= programa
 - ◆ \$3= parte1.c
 - ◆ \$4= parte2.c

Evaluación de variables

\$var

Valor de la variable *var*, si está definida

\${var}

Lo mismo pero delimita el nombre de la variable cuando está inserta en una cadena mayor

\${var=valor}

Valor de la variable *var*, si está definida. Si no se usa *valor*

\${var=valor}

Valor de la variable *var*, si está definida. Si no se usa *valor* y se asigna *valor* a *var*

\${var?mensaje}

Valor de la variable *var*, si está definida. Si no imprime *mensaje* y espera un valor para la variable proporcionado interactivamente

\${var+valor}

Usa *valor* si la variable *var* está definida

Selección de la shell

- ◆ Cada usuario tiene una shell por defecto para interactuar con el sistema operativo
- ◆ La shell por defecto está especificada en el último campo de la línea de registro del usuario en el fichero /etc/passwd
- ◆ El usuario puede cambiar su shell por defecto con chsh
 - Pide password y path de la nueva shell

Inicialización de la shell

- ◆ Cuando se ejecuta una shell interactiva, ésta ejecuta unos scripts de inicialización que asignan valores predeterminados por el usuario para variables especiales y de entorno y ejecutan comandos
- ◆ Estos ficheros de inicialización sirven para particularizar la sesión a los gustos y necesidades de cada usuario

Inicialización de la shell

- ◆ Los ficheros de inicialización de la shell residen en el directorio base del usuario
- ◆ También hay ficheros de inicialización globales para todos los usuarios
- ◆ Tienen nombres predeterminados
 - Inicialización de sesión para un usuario
 - ◆ \$HOME/.profile
 - Inicialización de sesión común para todos
 - ◆ /etc/profile

Tipos de Shell

- ◆ **Interactiva.** Permite interactuar al usuario con el sistema operativo
 - **Login shell:** La que se inicia tras el proceso de login. En un sistema con desktop gráfico, el propio desktop es la única login shell
 - **Non-login shell:** Una shell ejecutada como comando desde otra shell. Hereda el entorno y ejecuta ficheros de inicialización específicos. La shell de una ventana de terminal en Ubuntu es una non-login shell
- ◆ **No interactiva.** Entorno de ejecución de un comando lanzado por una shell interactiva, de la que hereda el entorno

Inicialización en bash

- ◆ Varios ficheros para particularización de comienzo y fin de sesión
 - /etc/profile
 - ◆ Inicialización global para login shells
 - /etc/basrc
 - ◆ Configuración global
 - \$HOME/.bash_profile
 - ◆ Ejecuta al comienzo de sesión
 - \$HOME/.bashrc
 - ◆ Ejecuta al comienzo de una shell
 - \$HOME/.bash_logout
 - ◆ Particulariza el fin de sesión

Ejemplo de .bash_profile

```
# .bash_profile

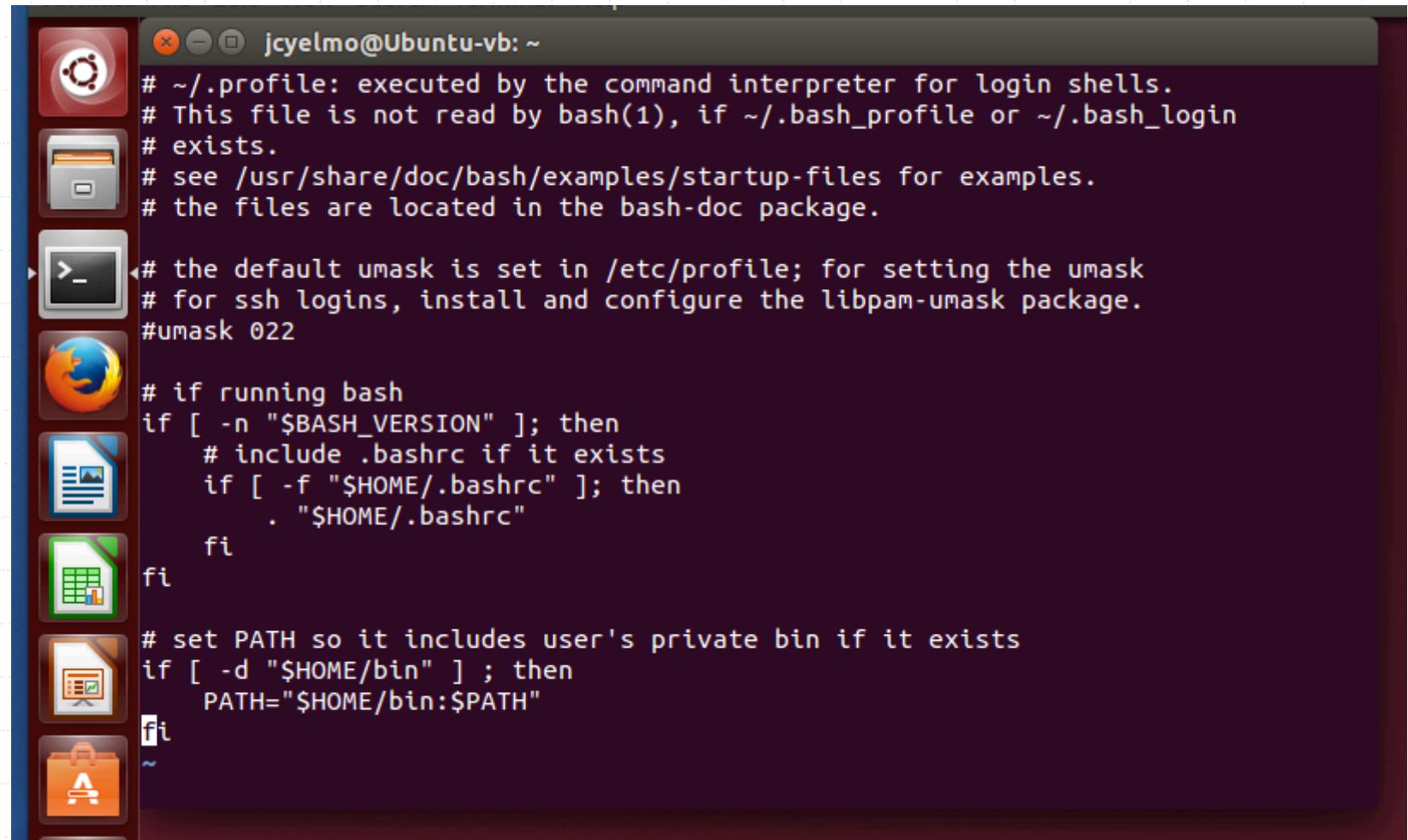
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin
BASH_ENV=$HOME/.bashrc
USERNAME=""

export USERNAME BASH_ENV PATH
```

Ejemplo de .profile



A screenshot of a Linux desktop environment, specifically Ubuntu, showing a terminal window. The terminal window title is "jcyelmo@Ubuntu-vb: ~". The window contains the content of a ".profile" shell script. The script includes comments about its execution by the command interpreter for login shells, its precedence over ".bash_profile" and ".bash_login", and examples from the bash-doc package. It sets the default umask to 022, checks if it's running bash, includes ".bashrc" if it exists, and sets the PATH to include the user's private bin directory if it exists. The terminal window has a dark theme and is surrounded by a grid pattern.

```
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
~
```



El sistema operativo UNIX

La interfaz de usuario III

Fin del tema

Juan Carlos Yelmo