Fundamentos de Software de Comunicaciones

Introducción al sistema operativo Linux

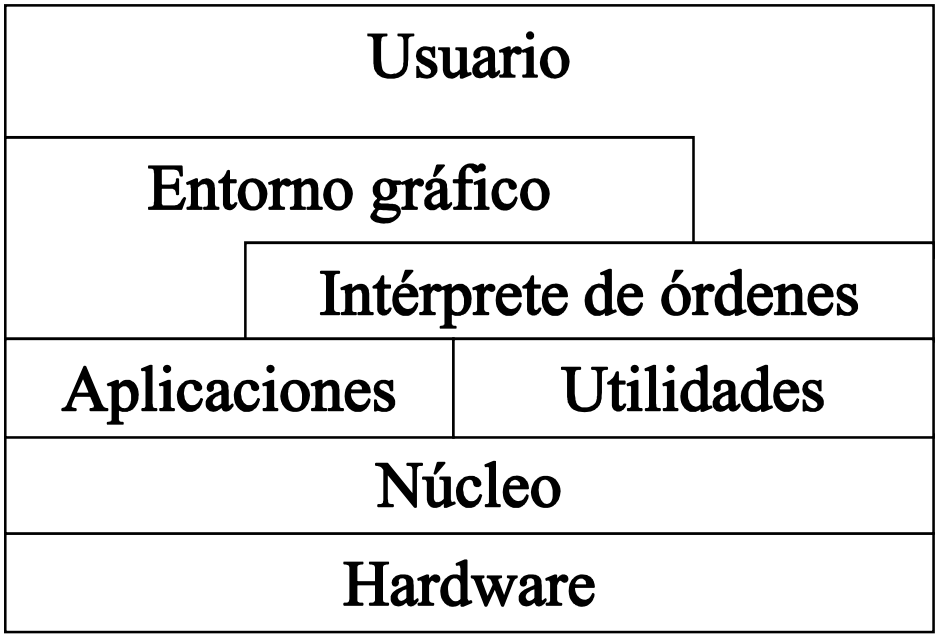
TUTORIAL CON EJERCICIOS

Características

- Linux es diseñado para ser un sistema operativo interactivo, multiusuario y multitarea:
  - Interactivo
    - El sistema acepta órdenes de los usuarios, las ejecuta y se dispone a esperar otras nuevas
  - Multitarea
    - Puede ejecutar varios trabajos de forma concurrente
  - Multiusuario
    - Más de una persona puede usar el sistema al mismo tiempo

Características

- La estructura interna de un sistema Linux se amolda a un típico esquema de capas



Trabajando con Linux

- La forma habitual de trabajo consiste en dar instrucciones al sistema a través de un terminal
- Tradicionalmente el terminal es de tipo textual
- En la actualidad se utilizan entornos gráficos
  - Aunque se sigue utilizando terminales para realizar gran número de tareas
  - El concepto de terminal se reemplaza por el de terminal virtual, que se ejecuta en una ventana

Trabajando con Linux

- El usuario interactúa con el sistema operativo utilizando un programa de utilidad denominado *shell*
- Existen varios:
  - Bourne shell (sh), C shell (csh), Korn shell (ksh), Bourne again shell (bash), etc

Trabajando con Linux

- Un shell es un intérprete de órdenes
- Es un programa que siempre se comporta de la siguiente forma:
  - Espera que el usuario introduzca una orden,
  - lee la orden del teclado, y
  - la ejecuta
- Para notificar al usuario que está dispuesto para aceptar una nueva orden
  - Se muestra un indicador (*prompt*): `%`, en el C shell, `$` en el Bourne shell

Trabajando con Linux

- Los shells son altamente configurables
  - Se pueden utilizar ficheros para realizar acciones predeterminadas al iniciar y terminar una sesión
- Incorporan un lenguaje propio
  - Con ellos se escriben programas, denominados *shell scripts*
  - Estos lenguajes son bastante potentes
  - Permiten realizar tareas complejas

Trabajando con Linux

- Los sistemas Linux proporcionan interfaces gráficas
  - La mayoría se basan en un sistema de ventanas distribuido que se denomina X-Window
- En la actualidad
  - Se apuesta por Gnome y KDE, dos entornos surgidos a partir de Linux

Introducción a la administración de sistemas

- En un sistema Linux hay dos tipos de usuarios
  - El administrador, conocido como super usuario o **root**
  - El resto de los usuarios
- El administrador tiene los máximos privilegios
  - El resto de usuarios tiene privilegios restringidos

Introducción a la administración de sistemas

- Qué hace cada tipo de usuarios
  - El super usuario tiene la función de administrar el sistema
  - Los usuarios pueden
    - Crear aplicaciones (programadores)
    - Usar aplicaciones ya existentes (usuarios finales)
- Los usuarios se organizan en grupos de trabajo
  - Los miembros de cada grupo tienen ciertos privilegios

Introducción a la administración de sistemas

- Funciones de administración
  - Mantener las cuentas de los usuarios
  - Gestionar y mantener el sistema de ficheros
  - Controlar el uso del sistema
  - Instalar nuevo software o actualizar el ya existente
  - Instalar y configurar nuevos dispositivos
  - Configurar el núcleo
  - Automatizar tareas repetitivas
  - Comprobar continuamente la seguridad del sistema

Acceso a un sistema Linux

- Se denomina sesión al período que transcurre desde el momento en que un usuario entra en el sistema hasta que lo abandona
- Para acceder a un sistema Linux un usuario debe disponer de
  - Un identificador de usuario
  - Una contraseña

Acceso a un sistema Linux

- El identificador de usuario y la contraseña son preguntados, leídos y comprobados (autenficados) por un programa llamado `login`
  - Esto se denomina *autentificación de usuarios*

Acceso a un sistema linux

- Los identificadores y las contraseñas
  - Se almacenan en un fichero (`/etc/passwd`) que contiene una línea por cada usuario
  - Cada línea contiene, entre otras cosas,
    - El identificador del usuario
    - La contraseña, que está encriptada
    - El shell por defecto del usuario
    - El directorio de conexión del usuario (directorio *HOME*)

Acceso a un sistema Linux

- Ejemplo de fichero `/etc/passwd`

```
root:x:0:1:Super-User:/:/bin/csh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
listen:x:37:4:Network Admin:/usr/net/nls:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
juanjose:x:1001:10:Juan Jose :/net/hercules/usuarios/juanjose:/bin/csh
antonio:x:1006:10:Antonio Nebro:/net/zeus/usuarios/antonio:/bin/csh
```

Acceso a un sistema linux

- Para terminar una sesión se puede usar las órdenes `exit`, `logout` o la combinación de teclas `CTRL-D`
- *Ejercicio*: inicie y termine una sesión en su sistema Linux

Acceso a un sistema Linux

- Las contraseñas se establecen y modifican con la orden `passwd`
- La elección de una contraseña ha de hacerse con determinados criterios
  - Debe tener una longitud mínima (entre seis u ocho caracteres, al menos)
  - No debe coincidir con el identificador de usuario
  - Debe ser difícil de averiguar por el resto de usuarios
  - Debe ser fácilmente recordable por el usuario

Acceso a un sistema Linux

- El administrador del sistema puede imponer algunas reglas
  - La longitud mínima
  - La obligatoriedad de incluir algún carácter numérico o especial
  - Las contraseñas pueden caducar
- *Ejercicio*: cambie la contraseña de su cuenta

Acceso a un sistema Linux

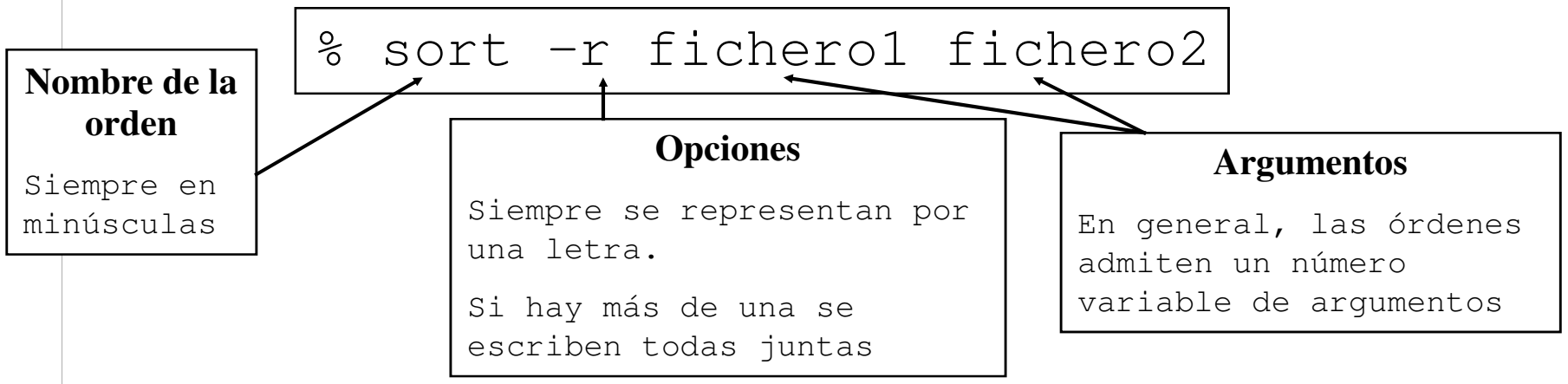
- El shell que se ejecuta al iniciarse una sesión se suele denominar *shell de conexión*
- A partir de un shell se pueden crear otros, pero el de conexión tiene ciertas propiedades que no tiene el resto

Acceso a un sistema Linux

- Cuando se inicial un shell de conexión
  - El shell asume como directorio por defecto el directorio HOME del usuario
- Directorio HOME (directorio personal)
  - Es el que le asigna el administrador al usuario como propio
  - Todo usuario tiene su propio directorio HOME

El Shell

- El shell es
  - El programa que actúa como intermediario entre el usuario y el sistema operativo
- Las órdenes que admite un shell suelen tener siempre el mismo formato:



El Shell

- Para introducir una instrucción
  - Basta escribirla después del indicador del shell y pulsar la tecla de retorno de carro
  - *Ejemplo*: ejecutar la orden `date`, que da la fecha y hora del sistema
  - *Ejemplo*: ejecutar la orden `clear`, que borra la pantalla
  - *Pregunta*: ¿qué ocurre cuándo una de estas instrucciones se escribe de forma incorrecta (por ejemplo, escribiendo el primer carácter en mayúsculas)?

El Shell

- Habitualmente todos los sistemas Linux tienen un manual en línea con información acerca de diversos aspectos del sistema
  - Órdenes ejecutables por los usuarios
  - Llamadas al sistema
  - Administración del sistema
  - Formatos de ficheros
- La orden para consultar el manual es `man`

El Shell

- Todo shell permite definir ciertos ficheros que pueden determinar las acciones a tomar cuando
  - Se inicia el shell de conexión
  - Se inicial un shell distinto del de conexión
  - Se cierra el shell de conexión

El Shell

- Estos ficheros son opcionales. En el C shell se denominan, respectivamente

  `.login`

  `.cshrc`

  `.logout`

- *Ejercicio*: comprobar si estos ficheros existen utilizando la instrucción `ls`, que lista el contenido de uno o varios directorios

El Shell

- Los ficheros cuyo nombre comienzan por el carácter punto, ‘.’
  - Se suelen utilizar para configurar las aplicaciones de cada usuario
- Al ser Linux un sistema multiusuario
  - Cada usuario puede tener unas determinadas preferencias
  - Éstas se almacenan en ficheros cuyo nombre comienza por punto, en el directorio HOME de cada usuario

El Shell

- Por defecto,
  - Los ficheros cuyo nombre comienza por punto no aparecen por defecto al usar la orden `ls`
  - Aunque eso no significa que sean ficheros ocultos

El Shell

- Si las hay, siempre van justo después del nombre de la orden
- Si hay varias, pueden ser separadas, precedidas del signo menos
- O pueden ir todas juntas

El Shell

- *Ejemplo*: estas dos instrucciones son equivalentes

```
% ls -a -l -F
% ls -alF
```

- *Ejercicio*: utilizando el manual, determinar para qué sirven las opciones `-F`, `-r` y `-t` de la instrucción `ls`

El Shell

- La mayoría de las instrucciones pueden tener un número variable de argumentos, que han de ir separados por espacios
  - *Ejemplo*:

```
% ls -lF .cshrc /usr/bin/ls
```

El Shell

- Hay que tener cuidado con algunos errores típicos
  - *Ejemplo*: las siguiente instrucción muestra las 5 primeras líneas de un fichero

    ```
    % head -5 /etc/passwd
    ```

  - *Pregunta*: ¿qué haría la instrucción en el siguiente caso?

    ```
    % head 5 /etc/passwd
    ```

El Shell

- Más errores típicos
  - *Ejemplo*: la siguiente instrucción borra todos los ficheros cuyo nombre acaba en "`.c%`"

    ```
    % rm *.c%
    ```

  - *Pregunta*: ¿qué haría la instrucción en el siguiente caso?

    ```
    % rm * .c%
    ```

El Shell

- El shell, como la mayoría de las órdenes de Linux, emplean tres ficheros denominados *ficheros estándar*
- Cuando un programa inicia su ejecución
  - tiene acceso de forma automática a tres ficheros llamados entrada estándar, salida estándar y error estándar (salida de errores)

El Shell

- Hay que tener en cuenta que en UNIX los dispositivos se tratan como ficheros
- Si no se indica lo contrario, los ficheros estándar son dispositivos
  - La entrada estándar es el teclado
  - La salida estándar es la pantalla
  - La salida de error estándar es la pantalla
- *Ejemplo*: ¿qué ocurre al ejecutarse lo siguiente?

```
% sort
```

El Shell

- Linux permite modificar los ficheros estándares de un proceso
  - Se utiliza el término *redirección de entrada/salida*
- Se puede redireccionar tanto la entrada como la salida estándar
  - La redirección de la salida de error es menos habitual

El Shell

- Se utiliza el carácter '>' o bien los caracteres '>>'
- *Ejemplos*: (hay que averiguar previamente para qué sirve la orden `cat`)

```
% ls -al / > listado
% cat listado
```

- Se puede usar para concatenar ficheros

El Shell

- *Ejemplos*:

```
% cat /etc/passwd > copia
% cat /etc/passwd listado > copia
% cat /etc/passwd listado >> copia
```

- *Pregunta*: ¿qué ocurre al ejecutarse la siguiente instrucción?

```
% cat > nuevo_fichero
```

El Shell

- Se utiliza el carácter ‘<’
- *Ejemplo*:

```
% cat < /etc/passwd
```

- *Pregunta*: ¿Qué diferencia existe entre la instrucción anterior y la que se muestra a continuación?

```
% cat /etc/passwd
```

El Shell

- Es posible redireccionar a la vez la entrada y la salida estándar
- *Ejemplo*:

```
% sort < /etc/passwd > computadores
```

El Shell

- Mediante redirección de entrada/salida se puede modificar la entrada o salida estándar
- Mediante tuberías (*pipes*) es posible conseguir que la salida estándar de una orden sea la entrada estándar de otra orden
  - Las tuberías permiten encadenar instrucciones
  - Se utiliza el símbolo '|'

El Shell

- *Ejemplo*: ¿qué se obtiene al ejecutar esta secuencia de instrucciones?

```
% who | wc -l
```

- *Ejercicio*: ¿qué se obtiene con esta otra?

```
% grep /etc/passwd | sort | head -15 |
  tail -5 > resultado
```

El Shell

- Aquellas órdenes cuya entrada y salida estándar se pueden redireccionar se denominan *filtros*
- Los filtros tienen la propiedad de que se pueden colocar en cualquier lugar dentro una secuencia de instrucciones encadenadas mediante tuberías

El Shell

- Ejemplos de filtros
  - `sort`
  - `cat`
  - `grep`
- Ejemplos de instrucciones que NO son filtros
  - `ls`
  - `lpr`

El Shell

- Los shells ofrecen la posibilidad de utilizar una notación abreviada para operar sobre conjuntos de ficheros y directorios en una única orden
- Esto se consigue mediante el uso de algunos caracteres que tienen significados especiales
- Estos caracteres se denominan *metacaracteres* y se emplean para establecer una correspondencia con nombres o partes de nombres de ficheros

El Shell

- Los más empleados son los siguientes:
  - Asterisco (*)
    - Representa cualquier cadena de caracteres, incluyendo la cadena vacía
  - Interrogación (?)
    - Representa a cualquier carácter simple
  - Corchetes ([])
    - Una lista de caracteres encerrada entre corchetes especifica que la correspondencia es con cualquier carácter simple de la lista
  - Guión (-)
    - El guión se utiliza dentro de los corchetes para indicar un rango de caracteres

El Shell

- *Ejercicio*: cree los siguientes ficheros
  - `a1, a11, a111, a2, aA, aB, aa, a10, a110, a12, a3, aA1, aG1, b1, b2, b3`
  - Y rellene la siguiente tabla con los ficheros seleccionados en cada caso

Referencia	Ficheros seleccionados
`a*`	
`a?`	
`a??`	
`a?*`	
`a?1`	
`[ab]*`	
`?1*`	
`a[A-Z]*`	
`[!a]*`	
`a[A-D]*`	
`?[1-9]*`	
`?[!1-2]*`	

El Shell

- Cuando el shell recibe una orden, por defecto se ejecuta en primer plano (*foreground*)
  - El shell ejecuta la orden y espera hasta que acabe
- Sin embargo, al ser Linux un sistema operativo multitarea, es posible ejecutar órdenes en segundo plano (*background*)
  - El shell ejecuta la orden, pero no espera a que ésta acabe
  - De esta forma se pueden ejecutar varias órdenes a la vez, de forma concurrente

El Shell

- La forma de indicar que se quiere ejecutar una orden en segundo plano es escribiendo al final de la misma el carácter '`&`'
- *Ejemplo*:

```
% ls -al / > listado &
```

  - ¿Qué ocurre si no se redirecciona la salida?

El Shell

- El entorno es un conjunto de elementos que establecen ciertas propiedades en la sesión de un usuario, como por ejemplo:
  - Dónde buscar los ficheros
  - Qué impresora usar por defecto
  - Qué aspecto tiene el indicador del shell

El Shell

- El entorno se configura habitualmente mediante lo se conoce como variables de entorno
  - Son pares `NombreDeVariable = Valor`
  - Se establecen habitualmente con la orden `setenv`
  - Se pueden observar mediante `printenv`
  - El valor de una variable de entorno se obtiene anteponiendo el carácter dólar (‘`$`’) al nombre de la variable
  - *Ejemplo*: `% echo $PATH`

Ficheros

- Definición de fichero
  - Unidad de información almacenada en disco
- Propiedades de los ficheros
  - La información que contienen la almacenan de forma persistente
  - Pueden ser compartidos por varios usuarios
  - Tienen un tamaño variable

Ficheros

- En Linux
  - Un fichero es una secuencia de bytes
  - Son las aplicaciones las que interpretan su contenido
  - Los ficheros tienen atributos
    - Longitud, propietario, grupo, fecha de último acceso, fecha de última modificación, permisos

Ficheros

- Los nombres de los ficheros pueden tener una longitud de hasta 256 caracteres. No pueden incluir el carácter ‘/’
- Las mayúsculas y minúsculas son distinguibles
  - Los nombres `ejemplo`, `EJEMPLO` y `Ejemplo` representan a tres ficheros distintos

Ficheros

- No existe el concepto de extensión en el nombre de un fichero
  - Son los usuarios y las aplicaciones los que usan el carácter punto como un separador para determinar los tipos de los ficheros. Ejemplos:

```
arbol.h      arbol.c       arbol.cpp
Arbol.java   Arbol.class   Arbol.class
arbol.tar    arbol.tar.Z   arbol.tar.gz
```

Ficheros

- Linux distingue tres tipos de ficheros
  - Ficheros ordinarios o regulares
    - Son secuencias de bytes agrupadas bajo un nombre
  - Directorios
    - Son ficheros que contienen listas de otros ficheros y directorios
    - Permiten estructurar los ficheros de un disco
  - Ficheros especiales o de dispositivo
    - UNIX trata los dispositivos como si fuesen ficheros

Ficheros

- No se distinguen entre ficheros de texto y ficheros binarios (todos son regulares)
- Sin embargo, algunos ficheros deben tener un formato interno concreto
  - Ejemplos: ficheros ejecutables, directorios
- Para saber el tipo de un fichero
  - Se puede usar `ls -l`

Ficheros

- *Ejemplos*:

```
% ls -l /etc/hosts
-rwxrwxrwx 1 root sys ... /etc/hosts
% ls -l /
drwxrwxrwx 1 root sys ... /etc
```

- El primer carácter del primer campo indica el tipo de fichero

Ficheros

- El primer carácter del primer campo puede tomar los valores siguientes:

Carácter	Tipo de fichero
`-`	Fichero regular
`d`	Directorio
`b`	Dispositivo de bloques
`c`	Dispositivo de caracteres
`l`	Enlace simbólico
`p`	Tubería (*pipe*)
`s`	Sockets

Ficheros

- Operaciones habituales sobre ficheros

Operación	Formas de llevarla a cabo
Crear	Editores, copiando, `cat`
Mostrar el contenido	`cat`, `more`, `head`, `tail`, `less`
Listar	`ls`
Copiar	`cp`
Renombar	`mv`
Borrar	`rm`
Imprimir	`lpr`
Enlazar	`ln`

Ficheros

- *Ejercicio*:
  - Cree un fichero llamado `hola.c` con la utilidad `cat`
  - El fichero deberá contener lo siguiente:

```
/* Fichero: hola.c */
#include <stdio.h>

int main(int argc, char** argv) {
  printf ("Esto es un programa C \n") ;
} /* main*/
```

Ficheros

- *Ejercicio*:
  - Para compilar el programa deberá usar una instrucción parecida a la siguiente:

```
% gcc hola.c –o hola
```

  - Ejecute el programa:

```
% hola
```

Ficheros

- *Ejercicios*:
  - Copiar el fichero `hola.c` a otro llamado `copia.c`
  - Compilarlo y obtener un fichero ejecutable llamado `copia`
  - Ejecutar el fichero `copia`
  - Comprobar que los ficheros `hola.c` y `copia.c` son iguales
  - Comprobar que los ficheros `hola` y `copia` son iguales

Ficheros

- *Ejercicios*:
  - Listar los ficheros con extensión "`.c`"
  - Borrar los ficheros `hola.c` y `hola` con una sola instrucción
  - La instrucción `file` da información sobre el fichero que recibe como parámetro. Utilizarla sobre los ficheros `copia.c` y `copia`
  - Determinar el tamaño en Kbytes de los dos ficheros
    - Con la instrucción `ls`
    - Con la instrucción `du` (*disk usage*)

Ficheros

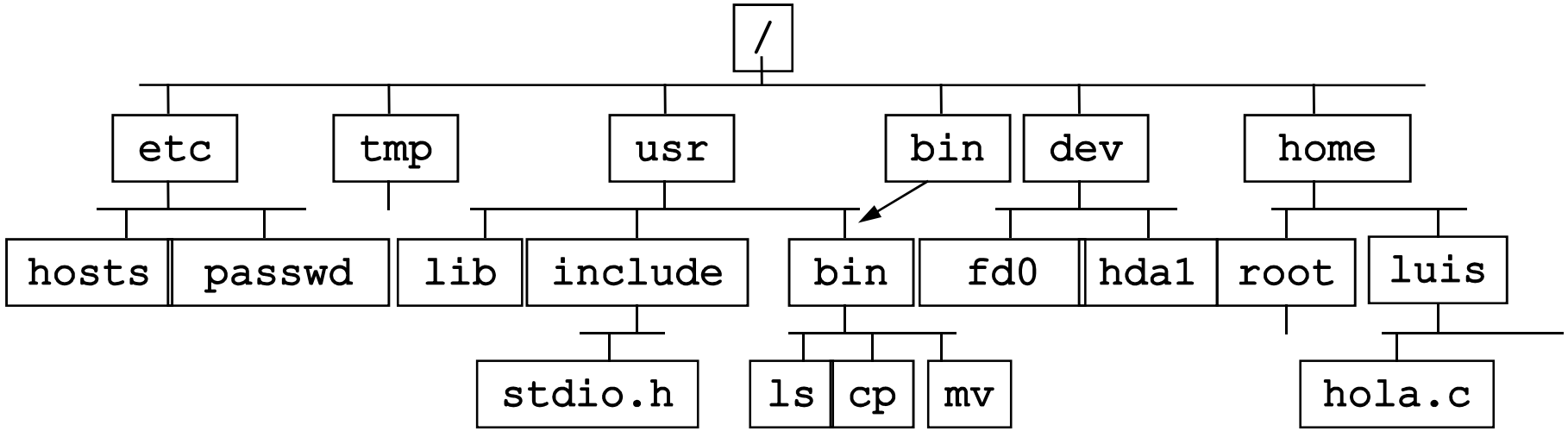
- *Ejercicios*:
  - Buscar la cadena "`main`" en el fichero `copia.c`
  - Obtener el número de la línea del fichero `copia.c` en la que se encuentra la cadena "`main`"
  - Contar las líneas, palabras y caracteres del fichero `copia.c`
  - Renombrar el fichero `copia.c` a `hola.c`

Directorios

- En Linux los directorios se almacenan en ficheros
- Las entradas de un directorio se pueden corresponder a ficheros u otros directorios
- Esto permite construir una jerarquía de directorios en forma de árbol
  - La raíz del árbol se conoce como directorio raíz, y se representa mediante el carácter ‘/’
- Todos los sistemas Linux
  - Tienden a organizar el árbol de directorios de la firma forma

Directorios

Jerarquía de directorios



Directorios

- Subdirectorio
  - Directorio que parte de otro directorio
- Directorio .
  - Referencia al propio directorio. Todo directorio tiene uno
- Directorio . .
  - Es el directorio padre de un subdirectorio. Todo directorio tiene uno

Directorios

- Enlace
  - Lugar al que hace referencia una entrada de directorio. Se representa mediante un *nodo índice*
- Nodo índice
  - Estructura de datos que utiliza UNIX para almacenar la información de un fichero o directorio

Directorios

- Directorio *HOME*
  - Directorio de conexión del usuario
- Directorio actual o de trabajo
  - Es el directorio que asume el shell por defecto
  - Se puede determinar mediante la instrucción `pwd`

Directorios

- Ruta (*path*)
  - Es la secuencia de directorios que se ha de seguir para llegar a un fichero o directorio concreto
  - La secuencia de directorios se separa por el carácter '/'
  - Ejemplos:

```
/usr/bin/ls
./hola
hola
/tmp
../../etc/passwd
```

Directorios

- Hay dos formas de hacer referencia a un fichero
- Direccionamiento absoluto
  - La ruta del fichero comienza por el directorio raíz
  - Hace referencia de forma unívoca al fichero desde cualquier sitio
- Direccionamiento relativo
  - La ruta del fichero NO comienza por el directorio raíz
  - La referencia al fichero dependerá del directorio actual

Directorios

- Ejemplos:

Direccionamiento absoluto

```
/usr/bin/ls
/tmp
/home/usuario/hola
```

Direccionamiento relativo

```
./hola
../../etc/passwd
hola
```

Directorios

- *Ejercicios*:
  - Desde el directorio *HOME*
    - Listar el contenido del directorio raíz mediante direccionamiento absoluto
    - Listar el contenido del directorio raíz utilizando direccionamiento relativo
    - Listar los ficheros del directorio `/dev` que empiecen por el carácter '`h`' utilizando direccionamiento absoluto y relativo

Directorios

- *Ejercicios*:
  - Se puede cambiar el directorio de trabajo con la orden `cd`
    - Cambiar el directorio de trabajo al directorio raíz. ¿Cómo se puede saber que realmente la orden se ha ejecutado con éxito?
    - Listar el contenido del directorio HOME desde el directorio raíz
    - Cambiar el directorio de trabajo al directorio HOME

Directorios

- Operaciones habituales sobre directorios

Operación	Formas de llevarla a cabo
Crear	`mkdir`
Mostrar el contenido	`ls`
Cambiar de directorio	`cd`
Copiar	`cp`
Renombar	`mv`
Borrar	`rmdir, rm`
Enlazar	`ln`

Directorios

- Un directorio se crea con la instrucción `mkdir`
- *Ejemplos*:

```
% cd
% mkdir temp
% mkdir -p programas/fuentes
```

Directorios

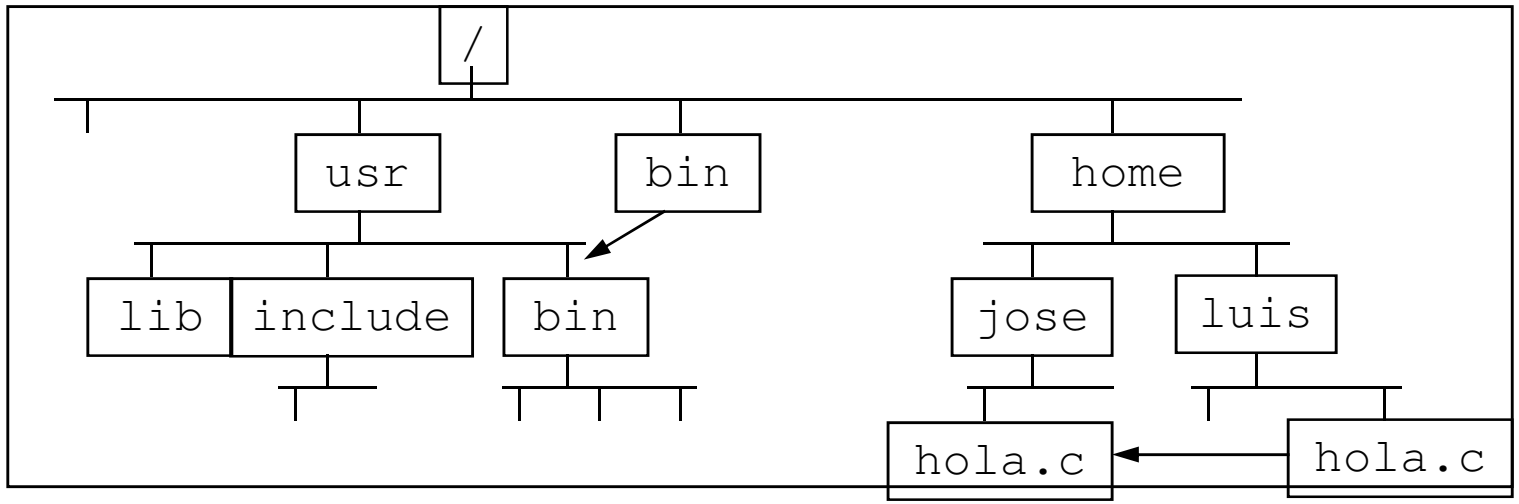
- Un directorio se borra con la instrucción `rmdir`
- *Ejercicio*: Borre el directorio `programas`. ¿Qué ocurre?
- *Ejercicio*: Borre todos los directorios creados con una sola instrucción
- Los directorios se pueden borrar completamente con la instrucción `rm -R`

Directorios

- En general, un enlace es una correspondencia entre una entrada de un directorio y el nodo índice del fichero al que dicha entrada hace referencia
- Sin embargo
  - Se suele hablar de enlaces en aquellas situaciones en las que más de una entrada de directorio hacen referencia a un mismo nodo índice
  - Cuando existe más de un enlace a mismo fichero o directorio, el árbol de directorios se transforma de un grafo

Directorios

Enlaces



Directorios

- Ventajas de utilizar enlaces
  - Se pueden compartir ficheros y directorios
  - Se pueden abreviar algunos direccionamientos cuya longitud es larga
- La instrucción para crear enlaces es `ln`

Directorios

- *Ejemplo*: cree un enlace al fichero `hola.c` llamado `enlace.c`

- *Pregunta*: ¿cómo se sabe que `hola.c` y `enlace` hacen referencia al mismo fichero?
- *Pregunta*: copie el fichero `enlace.c` a otro llamado `copia.c`. ¿Qué diferencia hay entre el fichero `enlace.c` y el fichero `copia.c`?
- *Pregunta*: borre el fichero `hola.c`. ¿Qué ocurre con el fichero `enlace.c`?

Directorios

- Enlaces a ficheros que se encuentran en otro directorio
  - *Ejercicio*: haga un enlace, llamado `mipasswd`, desde su directorio HOME al fichero `/etc/passwd`. Compruebe el número de enlaces del fichero `/etc/passwd`
- Enlaces a directorios
  - Sólo el superusuario puede hacer enlaces a directorios
  - *Pregunta*: ¿por qué?

Directorios

- Enlaces convencionales o “duros” (*hard links*)
  - Son referencias DIRECTAS a nodos índice desde un directorio
- Enlaces simbólicos
  - Son referencias INDIRECTAS a nodos índice desde un directorio
  - En realidad, son ficheros especiales que contiene una ruta
  - Esta ruta hace referencia al fichero enlazado

Directorios

- Ventajas de usar enlaces simbólicos
  - Es la única de forma de hacer enlaces a ficheros que se encuentran en dispositivos diferentes
  - *Ejercicio*: haga un enlace simbólico al fichero `hola.c` llamado `hola_s.c`
  - *Pregunta*: ¿cómo se sabe que `hola_s.c` es un enlace de `hola.c`?

Directorios

- Un inconveniente de usar enlaces convencionales
  - *Pregunta*: un usuario `A` hace un enlace a un fichero `f` del usuario `B`, sin que éste se percate de ello. ¿Qué ocurre cuando el usuario `B` borra el fichero `f`?
  - *Pregunta*: ¿ocurre lo mismo con los enlaces simbólicos?
  - *Conclusión*: es recomendable usar enlaces simbólicos para hacer enlaces a ficheros de otros usuarios

Directorios

- Inconvenientes de usar enlaces simbólicos
  - Ocupan un fichero adicional
  - La localización del nodo índice del fichero es más lenta
  - *Pregunta*: comprobar si se puede hacer un enlace simbólico al directorio raíz. ¿Qué ocurre entonces con las instrucciones que tienen un comportamiento recursivo?

Seguridad

- El término seguridad se aplica a
  - Todas aquellos aspectos relacionados con evitar que información de un sistema informático
    - Se pierda
    - No esté disponible a los usuarios
    - Sea utilizada por alguien que no está autorizado a hacerlo
  - Implica acciones de tipo administrativo, legal y técnico

Protección

- El término protección se puede aplicar a
  - Todos aquellos elementos que ofrece un sistema operativo en vistas a proporcionar seguridad
- Linux proporciona los siguientes mecanismos de protección
  - Autentificación de usuarios
  - Hay varios niveles de protección de los ficheros, de acuerdo con los distintos usuarios y grupos
  - Los espacios de direcciones de los procesos son privados

Protección

- Existen dos motivos fundamentales por los que hay que proteger los ficheros
  - Accesos malintencionados
  - Accesos accidentales

Protección

- Para garantizar que ficheros son accedidos de forma conveniente, UNIX distingue tres tipos de permisos y tres tipos de usuarios
  - Tipos de permisos
    - Permiso de lectura (`r`)
    - Permiso de escritura (`w`)
    - Permiso de ejecución (`x`)
  - Tipos de usuarios
    - Propietario del fichero (`u`)
    - Grupo del propietario (`g`)
    - Resto de usuarios (`o`)

Protección

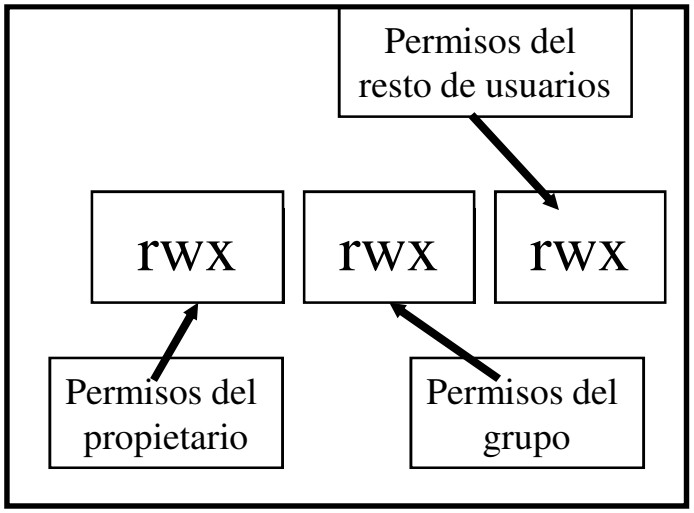
- El permiso de lectura (r) permite
  - Leer y copiar ficheros
  - Listar el contenido de los directorios
- El permiso de escritura (w) permite
  - Alterar o borrar el contenido de un fichero
  - Crear y borrar ficheros en un directorio
- El permiso de ejecución (x) permite
  - Ejecutar un fichero (si éste contiene código ejecutable)
  - En el caso de un directorio, se permite que un usuario se cambie al directorio (lo convierta en el directorio actual)

Protección

- Cada fichero tiene asociado una máscara de protección
  - Es una secuencia de bits, donde cada bit en la secuencia tiene un significado
- La máscara de protección de un fichero tiene tres grupos de tres bits
  - Cada grupo representa, respectivamente, los permisos del propietario, del grupo y del resto de usuarios

Protección

- Dentro de cada grupo
  - El primer bit representa el permiso de lectura
  - El segundo representa el permiso de escritura
  - El tercero representa el permiso de ejecución



Protección

- *Ejemplos*:

`rw- rw- rw-`

  - Todos los usuarios pueden leer y escribir el fichero

`rwx r-- r--`

  - El propietario tiene control total. El resto sólo puede leer el fichero

`r-- r-- r--`

  - Fichero de sólo lectura

`rwx rwx rwx`

  - Todos los usuarios tienen control total sobre el fichero

Protección

- *Pregunta*: ¿qué indican los siguientes permisos?

```
rwx r-x r-x

--- --- ---

rwx --- r-x
```

Protección

- Para determinar los permisos de un fichero
  - Una forma es mediante la instrucción `ls -l`
  - *Ejemplo*:

```
% ls -l hola.c
-rw-r--r--  1 antonio ...
```

Protección

- Notación de los permisos en base 8 (octal)

rw- rw- rw-	110 110 110	$666_{(8}$
rwx r-- r--	111 100 100	$644_{(8}$
r-- r-- r--	100 100 100	$444_{(8}$
rwx rwx rwx	111 111 111	$777_{(8}$

Protección

- Los permisos de un fichero se pueden cambiar con la instrucción `chmod`
  - *Ejemplos*: Comprobar que efectivamente se han cambiado los permisos después de cada instrucción

```
% chmod 777 hola.c
% chmod 644 hola.c
% chmod 400 hola.c
```

Protección

- *Pregunta*: ¿qué ocurre si se intenta borrar el fichero `hola.c` cuando sus permisos son 400?
- *Pregunta*: ¿qué ocurre si se intenta ejecutar la instrucción siguiente?

```
% chmod 777 /etc/passwd
```

Protección

- *Ejemplos*: Comprobar que efectivamente se han cambiado los permisos después de cada instrucción

```
% chmod u+x hola.c
% chmod go=r *.c
% chmod o-r,g=r hola.c
% chmod -R 644 *
```

Protección

- Permisos por defecto
  - *Ejercicio*: crear un nuevo fichero llamado `prueba.txt` con la orden `cat`. Comprobar qué permisos le ha asignado el sistema
  - *Ejercicio*: crear un nuevo directorio llamado `temp` y comprobar sus permisos

Protección

- Los permisos por defecto son
  - 666 para ficheros
  - 777 para directorios
- Sin embargo
  - Estos permisos se suelen modificar mediante la instrucción `umask`
  - *Ejemplo*:

```
% umask
022
% umask 044
```

Protección

- Propietario de un fichero
  - El propietario de un fichero se puede modificar mediante la instrucción `chown`
  - La instrucción `chown` sólo la puede ejecutar el superusuario
    - Por motivos de seguridad
    - Para evitar situaciones por uso malintencionado

Protección

- *Ejercicio*: comprobar que los ficheros y directorios del directorio HOME pertenecen al usuario
- *Ejercicio*: determinar quién es el propietario de los ficheros y directorios que cuelgan de directorio raíz

Protección

- Grupos
  - Los usuarios pueden pertenecer a uno o más grupos de trabajo
  - *Pregunta*: ¿cómo se puede saber a qué grupos pertenece un usuario?
  - *Pregunta*: ¿cómo se puede cambiar el grupo al que pertenece un fichero?

Procesos

- Un proceso es un programa en ejecución
  - Un proceso ejecuta una secuencia de instrucciones, resultado de la compilación de un programa escrito en un lenguaje de programación
- Un proceso es la entidad que puede tener actividad propia dentro del sistema operativo
  - El resto de entidades o recursos son pasivos
- Un proceso es la entidad a la que el sistema operativo proporciona recursos
  - Memoria, ficheros, dispositivos, tiempo de CPU

Procesos

- Los procesos tienen atributos
  - Identificador del proceso (PID)
  - Propietario (el usuario que lo creó)
  - Identificador del proceso padre (PPID)
  - Prioridad
  - Tiempo de procesamiento consumido
  - Estado (actividad, dormido, suspensión, *zombie*)

Procesos

- Al ser UNIX un sistema multitarea
  - Hay más de un proceso en ejecución
  - *Ejercicio*: ejecute la instrucción `ps -ef` para obtener todos los procesos del sistema
  - *Pregunta*: ¿qué significa cada elemento de las líneas de información que devuelve la orden anterior?

Procesos

- Operaciones básicas sobre procesos
  - Creación
  - Terminación
  - Suspensión
  - Reanudación
  - Cambiar prioridad

Procesos

- Dos formas básicas para crear procesos
  - El usuario ejecuta instrucciones por medio de un shell
  - Un proceso puede crear otros procesos

Procesos

- Los procesos se pueden ejecutar en primer plano o en segundo plano
  - En primer plano (*foreground*)
    - El proceso tiene como entrada estándar el teclado
    - Sólo puede haber un único proceso en primer plano
  - En segundo plano (*background*)
    - Puede haber más de un proceso
    - La entrada estándar de estos procesos no puede ser el teclado

Procesos

- Al ejecutar una instrucción en segundo plano
  - El shell devuelve un identificador de trabajo (*job*) y el identificador de proceso (PID) que se ha generado
  - *Ejemplo*:

```
% df > info.txt &
[1] 3254
```

Procesos

- Los procesos generados al introducir el usuario instrucciones por medio del shell suelen tener una duración corta
- Algunas aplicaciones suelen llevar mucho tiempo
  - Procesamiento de imágenes
  - Programas de cálculo científico
- Existen procesos del sistema denominados *daemons* (demonios) que se crean cuando arranca el sistema operativo y están vivos hasta que el sistema es apagado

Procesos

- Cuando un proceso termina por sí solo
  - El shell indica este hecho mostrando en pantalla el PID del proceso que ha acabado
- Los procesos se pueden terminar explícitamente, con la instrucción `kill`
  - *Ejemplo*: una forma es utilizando como argumento el PID

```
% kill 3254
```

Procesos

- *Ejemplo*: otra forma de terminar un proceso es utilizando como argumento el identificador del trabajo

```
% kill %1
```

- *Pregunta*: qué ocurre al intentar eliminar el proceso generado al ejecutar la instrucción

```
df > info.txt &
```

Procesos

- Ejemplo de proceso de larga duración: buscar ficheros
  - *Ejercicio*: ejecute la siguiente instrucción, que busca todos los ficheros del sistema de cuyo nombre empieza por la cadena "`pa`", en segundo plano

    ```
    % find / –name "pa*" –print > pas.txt &
    ```

  - *Ejercicio*: utilice la instrucción `kill` para eliminar el proceso

Procesos

- Si se ejecuta de nuevo la instrucción anterior, hay dos formas para saber que el proceso está en ejecución
  - Con la instrucción `ps`
  - Con la instrucción `jobs`
  - *Ejercicio*: compruébelo

Procesos

- Formas de eliminar un proceso en primer plano
  - Enviándole una señal para que termine, pulsando `CONTROL-C`
  - Con `kill`, durmiendo previamente el proceso
    - Un proceso se duerme pulsando `CONTROL-Z`
  - *Ejercicio*: compruébelo con la instrucción

```
% find / -name "pa*" -print > pas.txt
```

Procesos

- Una señal es un mecanismo por el que se le notifica a un proceso la ocurrencia de un suceso asíncrono
  - Cuando un proceso recibe una señal es interrumpido para poder tratarla
- Las señales se envían desde el shell usando la instrucción `kill`

Procesos

- El efecto que produce una señal sobre un proceso depende de la señal
  - La señal se puede ignorar
  - El proceso es terminado
  - El proceso es suspendido
  - El proceso es despertado
- Para saber las señales que define un sistema Unix se usa `kill -l`. Compruébelo

Procesos

- *Ejemplos*:
  - `kill -2 pid` envía la señal de interrupción
    - El mismo efecto se consigue con `CONTROL+C`
  - `kill -9 pid` envia la señal de terminación incondicional
  - `kill -STOP pid` duerme un proceso
    - El mismo efecto se consigue con `CONTROL+Z`
  - `kill -CONT pid` despierta un proceso previamente dormido

Procesos

- El C shell permite
  - Conmutar la ejecución de un proceso de primer plano a segundo plano y viceversa
  - Instrucciones: `jobs`, `fg`, `bg`
  - *Ejercicio*: ejecute la instrucción que busca los ficheros cuyo nombre empieza por "`pa`" en primer plano y haga el proceso generado pase a segundo plano
  - *Ejercicio*: haga ahora que pase de segundo plano a primer plano

Procesos

- Linux es un sistema operativo que asigna propiedades a los procesos
  - Cuando tiene que ejecutar un proceso, elige siempre al que tiene mayor prioridad
- Los procesos de usuario siempre tienen una prioridad por defecto
- El usuario puede cambiar esta prioridad
  - Pero únicamente para reducir la prioridad del proceso
  - Sólo el super usuario puede aumentar la prioridad de un proceso

Procesos

- *Ejercicio*: consultando el manual, averigüe cuál es la orden (u órdenes) para cambiar la prioridad de un proceso, y compruebe su funcionamiento