

---

# Sistemas de Ficheros en GNU/Linux

# *Nota de Copyright*

---

© 2005 Diego Chaparro. Algunos derechos reservados.

Este trabajo se distribuye bajo la licencia Creative Commons Attribution-ShareAlike. Para obtener la licencia completa, véase <http://creativecommons.org/licenses/by-sa/2.1/es>

# *Particiones*

---

- ◆ Un disco se divide en secciones llamadas particiones. Debe tener un mínimo de 1 partición y un máximo de 4 particiones primarias
- ◆ En lugar de 4 primarias podemos crear 3 primarias y una extendida. Dentro de la partición extendida creamos las particiones lógicas
- ◆ Las particiones son formateadas para crear el sistema de ficheros del SO que usemos

# Particiones

## ◆ Nombres de partición

◆ el nombre de una partición está compuesto por varios caracteres:

- ◆ Tipo de controlador (h: IDE, s: SCSI)
- ◆ Tipo de dispositivo: d: disk
- ◆ Número de disco: a (primero), b(segundo), c(tercero), ...
- ◆ Número de partición

◆ Ejemplos:

- ◆ /dev/hda2      Controlador IDE, primer disco, 2ª partición
- ◆ /dev/sdb3      Controlador SCSI, segundo disco, 3ª partición

# Particiones

## ◆ ¿Cuántas necesitamos?

- ◆ Pues como mínimo necesitamos dos particiones:
  - ◆ / Es el sistema de ficheros completo del sistema
  - ◆ swap Se crea una partición para memoria virtual
- ◆ Con esto es suficiente para que funcione el sistema. La swap se utiliza para almacenar páginas de memoria no usadas
- ◆ Siempre se ha dicho que la swap debe ser más o menos del tamaño de la RAM. Pero realmente la swap ralentiza el sistema porque la paginación a disco es muy lenta. Por tanto podemos dar el tamaño que queramos a la swap, pero si nuestro sistema usa mucho la swap, esto significa que debemos aumentar la RAM

# Particiones

## ◆ ¿Cuántas necesitamos?

- ◆ Pero es recomendable hacer varias particiones, por las razones siguientes:
  - ◆ Si una partición falla, las demás particiones no se ven afectadas
  - ◆ Si tienes que formatear una partición, las otras no es necesario tocarlas
  - ◆ Tenemos una partición para almacenar los datos que pueden crecer constantemente (logs, correo, ...), para evitar que dejen el sistema sin disco en un sistema sin esta partición
  - ◆ El tiempo de chequeo de un sistema de ficheros es menor

# Particiones

## ◆ ¿Cuántas necesitamos?

◆ Un diseño de particiones para un sistema podría ser:

- ◆ / Contiene aplicaciones y ficheros de configuración
- ◆ /var Contiene ficheros de log, correo, www
- ◆ /usr Contiene el software instalado
- ◆ /tmp Ficheros temporales
- ◆ /home Directorios de usuario
- ◆ swap Memoria virtual

# Particiones

## ◆ fips

- ◆ Si vamos a instalar un equipo que no tiene espacio libre para crear una partición, podemos usar fips.exe para dividir una partición existente en 2
- ◆ Esta partición debe ser de tipo FAT16
- ◆ Lo que hace fips es reducir la partición que le especifiquemos y deja espacio libre para poder crear otra
- ◆ Podemos descargarlo de:  
<http://sunsite.unc.edu/pub/Linux/system/install>
- ◆ Y también lo podemos encontrar en los CDs de instalación de debian por ejemplo, en el directorio *tools*
- ◆ Es conveniente realizar un backup antes de usarlo

# Particiones

## ◆ fdisk

- ◆ Nos permite manipular la tabla de particiones
- ◆ Podemos crear particiones, borrarlas, cambiarles el tipo
- ◆ Sintaxis: *fdisk disco*
- ◆ Podemos interactuar con él mediante comandos:
  - ◆ a Marca la partición como bootable
  - ◆ d Borra partición
  - ◆ m Ayuda
  - ◆ n Añade nueva partición
  - ◆ p Muestra la tabla de particiones
  - ◆ q Sale sin grabar los cambios

# Particiones

- ◆ El sistema de ficheros en Linux organiza archivos y directorios en forma de árbol
- ◆ Algunos sistemas de ficheros:
  - ◆ ext (Extended Filesystem): después ext2 y ahora ext3
  - ◆ ISO9660 Sistema de ficheros de los CDROM
  - ◆ minix El primer sistema de ficheros usado por Linux
  - ◆ FAT16 Sistema de ficheros de MS-DOS
  - ◆ NTFS Sistema de ficheros de Windows NT
  - ◆ proc Sistema de ficheros virtual que proporciona información
  - ◆ VFAT Extensión de fat16
  - ◆ REISERFS, XFS, JFS

# Particiones

## ◆ Sistemas de ficheros con Journaling:

- ◆ Básicamente aumentan la consistencia del sistema de ficheros
- ◆ Para ello van guardando la meta-información (inodos nuevos, bloques liberados, ...) en un log
- ◆ Si el sistema falla y no se han volcado los datos de cache a disco, el sistema de ficheros lee el fichero de log y hace que el sistema de ficheros sea consistente
- ◆ ReiserFS      Mejor para leer ficheros a medianos
- ◆ XFS            Mejor para ficheros grandes

# Particiones

## ◆ Formatear

- ◆ Para formatear y crear el sistema de ficheros: `mkfs -t fs`
- ◆ En realidad llama a otros comandos dependiendo del tipo que especifiquemos:
  - ◆ `mkfs.ext2`
  - ◆ `mkfs.msdos`
  - ◆ `mkfs.minix`
- ◆ El número de bloques se puede ver con el `fdisk`. Pero yo creo que no es necesario especificarlo
- ◆ Opciones:
  - ◆ `-c` Chequea en busca de bloques defectuosos
- ◆ Para formatear disquetes se suele usar `fdformat`: `fdformat /dev/fd0`

# Organización

## ◆ Organización de directorios:

- ◆ / Directorio raíz
- ◆ /boot Arranque del sistema
- ◆ /bin Binarios
- ◆ /dev Ficheros de dispositivos, periféricos
- ◆ /etc Ficheros de configuración
- ◆ /home Directorios de usuarios
- ◆ /lib Librerías compartidas
- ◆ /mnt Usado para montar particiones temporales
- ◆ /proc Información sobre el kernel y procesos

# Organización

## ◆ Organización de directorios:

- ◆ /tmp Ficheros temporales de aplicaciones
- ◆ /usr Software, documentación, ...
  - ◆ /usr/src Aquí están los fuentes del kernel
- ◆ /var Ficheros de log, ficheros que sirve apache, correos, ... (el contenido de este fichero puede variar mucho)
  - ◆ /var/spool/mail/ Se guardan los e-mails de los usuarios
  - ◆ /var/www/ Páginas que sirve apache

# Organización

## ◆ Inodos

◆ Estructura de datos que almacena la información sobre cada fichero:

- ◆ Puntero al fichero físico
- ◆ Nombre de fichero
- ◆ Propietario y grupo ids
- ◆ permisos
- ◆ tamaño
- ◆ fecha último acceso
- ◆ número de links al archivo, ...

## ◆ **stat**

◆ Muestra la información sobre el inodo de un fichero

# Organización

## ◆ Ficheros:

- ◆ Hay varios tipos de ficheros
- ◆ Podemos ver el tipo en el primer carácter del bloque de permisos:
  - Fichero ordinario
  - b Dispositivo de bloques (disco duro, disquetera, ...)
  - c Dispositivo de caracteres (impresora, ...)
  - d Directorio
  - l Enlace

# Mantenimiento

## ◆ fsck

- ◆ Para mantener el sistema de ficheros sin journaling es necesario chequear el sistema de ficheros para comprobar su integridad
- ◆ Sintaxis: `fsck -t fs_tipo dispositivo`
- ◆ Ejemplo: `fsck -t ext2 /dev/hda1`
- ◆ Para chequear la partición / es necesario desmontarla, como no podemos porque la estamos usando la solución sería montarla como solo lectura, chequearla al principio o arrancar con un disquete y chequearla
- ◆ Opciones:
  - a Funciona en modo no interactivo
  - c Chequea por si hay bloques defectuosos

# Mantenimiento

## ◆ du

- ◆ Devuelve el espacio ocupado por un fichero o directorio recursivamente

- ◆ *du [opciones] [fichero]*

- ◆ Opciones:

- a Muestra el tamaño de los ficheros
- b Muestra el resultado en bytes
- h Muestra las medidas en forma más intuitiva

# Mantenimiento

## ◆ df

◆ Devuelve la información sobre el tamaño de dispositivos como particiones

◆ Sintaxis: *df [options] [fichero]*

◆ Devuelve:

- ◆ Tamaño del dispositivo
- ◆ Número de bloques libres
- ◆ Número de bloques ocupados
- ◆ Porcentaje de espacio libre
- ◆ Punto de montaje

◆ Opciones:

- h Muestra información sobre las unidades de medida utilizadas
- m Muestra en megabytes

# *Mantenimiento*

## ◆ **Montaje de particiones**

- ◆ Para acceder al sistema de ficheros, primeros necesitamos hacer accesibles esas particiones para poder leer los ficheros que hay en ellas
- ◆ A eso es a lo que se se le llama montaje, se puede montar un disquete, disco, sistema de ficheros virtual (/proc), ...
- ◆ Montar una partición significa hacer accesible ese dispositivo desde algún punto de nuestra jerarquía de directorios

# Mantenimiento

## ◆ mount

◆ Nos permite montar dispositivos en un punto de montaje

◆ Sintaxis: *mount [opciones] [dispositivo] punto\_montaje*

◆ Opciones:

- a Monta todos los sistemas de ficheros de /etc/fstab
- f Chequea los sistemas que se pueden montar, pero no los monta
- n No escribe información de montaje en /etc/mstab
- o Modifica el montaje
- r Monta como solo lectura
- t Especifica el tipo de sistema de ficheros
- v Muestra información sobre lo que hay mpointado
- w Monta como lectura-escritura

# *Mantenimiento*

## ◆ **mount**

### ◆ Ejemplos:

- ◆ `mount -t iso9660 /dev/hdb /cdrom`
- ◆ `mount -t ext2 /dev/hda2 /home`

# Mantenimiento

## ◆ mount

### ◆ Opciones para el -o

ro	Montar como solo lectura
rw	Montar como lectura escritura
users	Permite a los usuarios montar el sistema de ficheros
remount	Monta un sistema de ficheros ya montado
loop	Monta una imagen de disco en un dispositivo de loop

### ◆ Ejemplos:

- ◆ `mount -o remount,rw /`
- ◆ `mount /home/pepe/cd.iso /mnt -t iso9660 -o loop=/dev/loop0`
- ◆ `mount /tmp/fdimage /mnt -t msdos -o loop=/dev/loop3,blocksize=1024`

# Mantenimiento

## ◆ /etc/fstab

- ◆ En este fichero se definen las particiones a montar y desmontar. También especifica que sistemas de ficheros hay que chequear (fsck)
- ◆ Sintaxis:
  - ◆ Dispositivo a montar
  - ◆ Punto de montaje
  - ◆ Tipo del sistema de ficheros
  - ◆ Opciones de montaje
  - ◆ Indica si hacer el backup o no al usar dump (ya lo veremos)
  - ◆ Orden en el que queremos chequear con fsck

# Mantenimiento

## ◆ /proc

- ◆ Es un sistema de ficheros virtual, que nos muestra información sobre el sistema y sobre los procesos en ejecución
- ◆ Cada proceso en ejecución tiene un directorio con su pid, y contiene:
  - ◆ cmdline                      Línea de comando del proceso
  - ◆ cwd                              Enlace al directorio de trabajo del proceso
  - ◆ environ                        Entorno del proceso. Para verlo: `cat environ | tr "\000" "\n"`
  - ◆ exe                              Enlace al ejecutable del proceso
  - ◆ fd                                Directorio con los fd(File Descriptor) usados por el proceso
  - ◆ map                             Información sobre la memoria usada por el proceso
  - ◆ root                            Indica el / del proceso, por el chroot
  - ◆ stat                             Información de estado sobre el proceso
  - ◆ status                         Más información de estado

# *Mantenimiento*

## ◆ **/etc/mtab**

- ◆ Contiene información sobre los sistemas de ficheros montados
- ◆ Es lo que se muestra al ejecutar mount sin argumentos
- ◆ Es creado al arrancar

## ◆ **umount**

- ◆ Desmontamos un sistema de ficheros. En ese momento ya no podemos acceder a los ficheros del mismo
- ◆ Siempre hay que desmontar los dispositivos antes de quitarlos del sistema (disquetes, CDs, ...) para que se vacíen las cachés a los dispositivos
- ◆ Opciones:

# Enlaces

- ◆ Los enlaces se crean para no tener que duplicar los ficheros en el disco
- ◆ Dos tipos de enlaces: duros y simbólicos (o blandos)
  - ◆ **Enlaces duros**
    - ◆ Crea un nuevo puntero a un fichero
    - ◆ Todos los atributos de los enlaces son iguales
    - ◆ Si borramos uno de los enlaces el fichero sigue estando allí
    - ◆ Con `ls -l` vemos el número de enlaces que tiene
    - ◆ No importa qué enlace se creó primero, son iguales
    - ◆ Limites:
      - ◆ No se pueden enlazar directorio (solo root con la opción `-d` o `-F`)
      - ◆ Los enlaces tienen que estar en el mismo sistema de ficheros
    - ◆ Ejemplos:
      - ◆ `ln fichero nuevoenlace`

# Enlaces

- ◆ Dos tipos de enlaces: duros y simbólicos (o blandos)
  - ◆ **Enlaces simbólicos**
    - ◆ Simplemente es un puntero a un fichero existente
    - ◆ Permite:
      - ◆ Enlazar directorios
      - ◆ Enlazar a ficheros no existentes
      - ◆ Enlazar a otro sistema de ficheros
    - ◆ Ejemplo:
      - ◆ `ln -s fichero nuevoenlace`
    - ◆ `ls -l` nos muestra información sobre en enlace
    - ◆ Podemos borrar el fichero original, y entonces el enlace no valdrá porque apuntará a ningún sitio
    - ◆ `-F` para crear enlaces a directorios. El funcionamiento depende de cada shell

# Buscar Ficheros

## ♦ find

♦ Busca ficheros en un directorio dado recursivamente

♦ Sintaxis: find [path] [condicion]

♦ Condiciones:

-atime +n	<i>n</i> número de días desde el último acceso
-group nombre	Grupos de nombre <i>nombre</i>
-inum n	Número de inodo <i>n</i>
-links n	Número de links <i>n</i>
-mtime n	Número de días desde la última modificación
-name patrón	El nombre coincida con el patrón
-type c	Tipo igual a <i>c</i> [b,c,d,l,f]
-user nombre	Ficheros que pertenecen al usuario <i>nombre</i>

# Buscar Ficheros

## ◆ locate

- ◆ En lugar de buscar en el árbol de directorios busca en una base de datos con todos los ficheros del sistema
- ◆ Esta base de datos se actualiza regularmente
- ◆ La base de datos está en `/var/lib/dlocate/`
- ◆ Sintaxis: `locate [fichero]`
- ◆ Se puede pasar una expresión regular
- ◆ Ejemplo:
  - ◆ `locate *.tar`

# *Buscar Ficheros*

## ◆ **which**

- ◆ Busca en el path por el comando buscado y devuelve su ruta
- ◆ Es útil para saber cuál es exactamente el comando que estamos utilizando