Capítulo 2

Sistema de arquivos

Unha das funcións de calquera sistema operativo é xestionar a información almacenada no ordenador, nos distintos soportes físicos. Á parte do sistema operativo que se encarga desta tarefa denominámoslle *sistema de arquivos*.

En Linux opera a máxima de que "todo é un arquivo", non so os arquivos almacenados físicamente no disco duro, senón tamén as fontes de datos accesibles remotamente mediantes canles de comunicación como poden ser os *sockets*, as propias carpetas que conteñen arquivos, e mesmo os propios dispositivos fisicos (discos, impresoras, teclado ...).

Para poder xeralizar deste xeito Linux emprega un concepto que é importante entender, o de "sistemas de arquivos virtual ou *virtual file system* (VFS).

2.1. Sistema de arquivos virtual

Cando un programa de ordenador precisa acceder a un arquivo simplemente indica o nome do arquivo (mais ben o *path*) e a operación que quere facer nel (ler, escribir, borrar, etc). De feito os programas non acceden directamente aos discos senón que fan *chamadas* ao *kernel* do sistema, que é o único cos permisos precisos para facelo.

Para que o *kernel* non teña que preocuparse dos detalles "físicos" (se o arquivo está no disco duro, nunha memoria USB ou mesmo nun servidor de disco remoto) comunícase co VFS, tal e como se amosa na figura 2.1. O VFS é unha capa que ofrece unha API (*Application Program Interface* ao *kernel*) con funcións como crear arquivo, escribir nun arquivo, borralo, etc.

Físicamente os arquivos estarán nun soporte *hardware*, en algún sector dun

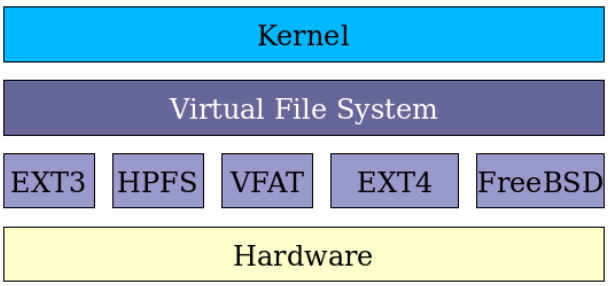

Figura 2.1: Esquema do sistema de arquivos virtual (sacado de [1])

disco. Para que o VFS poda "traducir" cada operación solicitada polo *kernel* á correspondente instrucción da cabeza lectora do disco precísase outra capas mais, tal e como se indica na figura 2.1, pero diso falaremos nas seccións seguintes.

Neste punto o importante é fixarse en que o VFS ofrece ao kernel a imaxe dun sistema de arquivos unificado. Se facemos o simil cun sistema público de bibliotecas sería como dispoñer dun *mapa* de libros unificado, no cal o lector poderia buscar, localizar un libro e pedilo, sin importarlle en que biblioteca concreta está, nin como organiza os libros nesa biblioteca o seu bibliotecario.

2.2. Estrutura do sistema de arquivos

En Linux todos os arquivos accesibles mediante o VFS estructúranse nunha única estrutura arborescente. A raíz *root directory* desa árbore represéntase co símbolo /. Esa carpeta conterá unha serie de carpetas e arquivos, e cada unha desas carpetas será o punto de partida dunha nova árbore.

Dende o ano 1993 existe un estándar que nos indica como debería ser esa estrutura, que carpetas deberán existir nese primeiro nivel, para que se debería empregar cada unha delas, que permisos deberían ter, etc. Ese estándar denomínase FHS *(Filesystem Hierarchy Standard)* e foi actualizado no ano 2004 [2].

Na figura 2.2 pódese observar a estructura que propón dito estándar, e que imos explicar a continuación. Hai moitos tuturais en internet explicando esta estructura tanto en inglés [4] como en castelán [5].

[1] http://opensource.com/life/16/10/introduction-linux-filesystems
[2] http://www.pathname.com/fhs/
[3] https://i.imgur.com/02VycVd.png
[4] https://www.linux.com/tutorials/linux-filesystem-explained/
[5] http://recursostic.educacion.es/observatorio/web/ca/software/software-general/493-sagrario-peralta

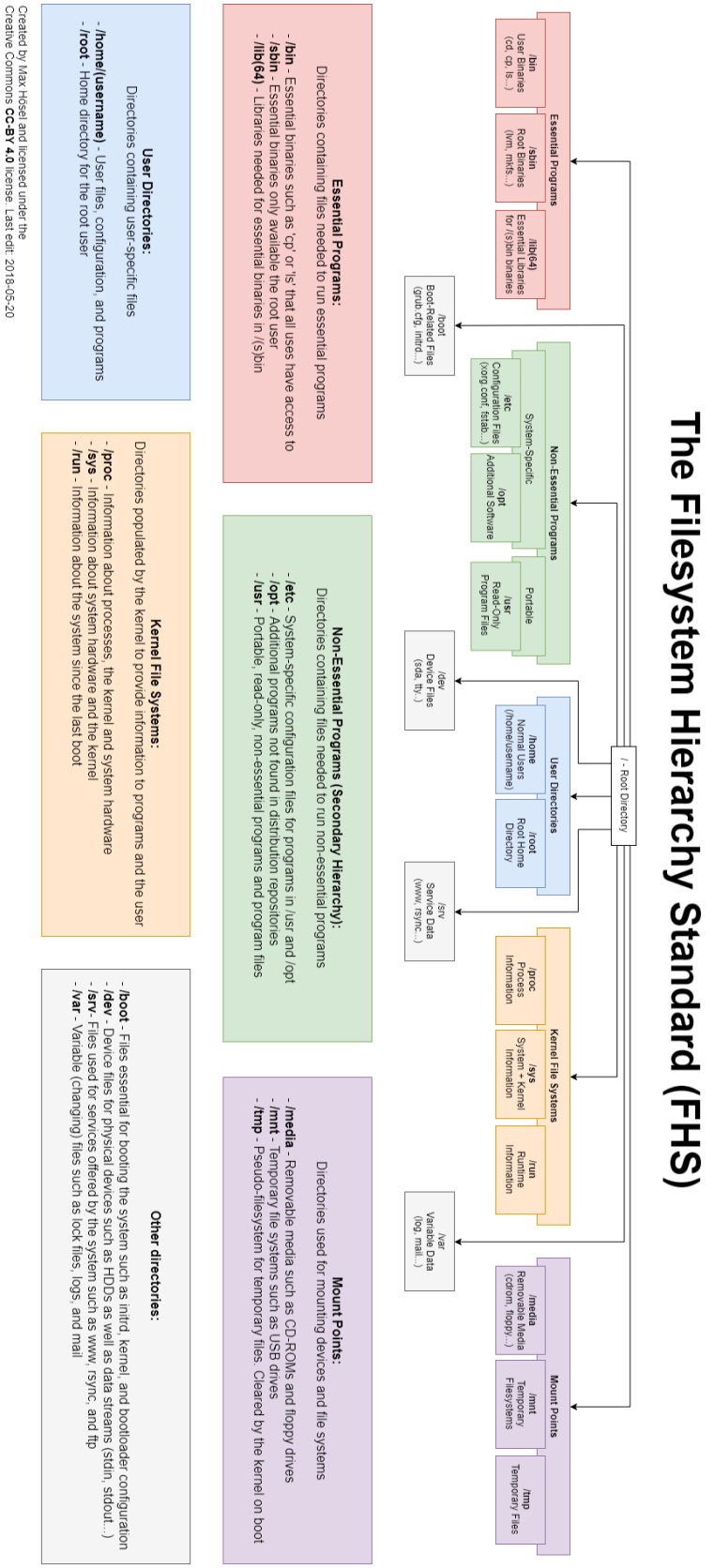

Figura 2.2: Estrutura de arquivos proposta polo FHS (obtido de [3])

- */bin:* contén o código binario dos arquivos executables (*bin* ven de *binary*) por todos os usuarios. Non debe ter subcarpetas. Exemplos de arquivos desta carpeta serían: *bash, cat, grep, more, sed* ou *tar*.

- */sbin:* contén os executables de *administración*, pensados para que os execute o usuario *root*. Exemplos de arquivos nesta carpeta serían: *fsck, iptables, mkfs* ou *shutdown*.

- */lib:* contén as librerías compartidas, empregadas por diferentes programas. Tamén contén os módulos do *kernel*, que estarán na subcarpeta */lib/modules*.

- */boot:* contén arquivos necesarios para o arranque do sistema. Como mínimo conterá o *kernel* do sistema pero tamén información da configuración do xestor de arranque (habitualmente *grub*).

- */etc:* esta carpeta contén arquivos de configuración. Exemplos de arquivos que atoparemos nesta carpeta son *crontab, hosts, passwd* ou *profile*. Soe haber moitas subcarpetas correspondentes a distintas aplicacións como por exemplo *bluetooth, cron.d, grub.d* ou *rc2.d*.

- */opt:* esta é a carpeta reservada para a instalación de aplicaciones adicionales, fora das esenciais do sistema operativo. Por exemplo no meu ordenador teño, entre outras, as seguintes subcarpetas *Adobe, brother* ou *google*.

- */usr:* esta carpeta é case unha segunda árbore de directorios, dentro da raíz. Segundo o estándar debería conter datos que podan compartirse con outros *hosts*, e de so lectura. Habitualmente emprégase tamén para instalar software no sistema. Soe conter subcarpetas como *bin, sbin, lib* ou *src*.

- */dev:* contén as definicións de todos os dispositivos conectados ao sistema. O sistema non poderia por exemplo ler datos do teclado, imprimir un arquivo ou escoitar o micrófono sen un arquivo para cada un deses dispositivos nesa carpeta. A información desta carpeta xenérase durante o arranque do sistema, ou en vivo, cando se conecta un dispositivo ao mesmo. Exemplos de arquivos que atoparemos nesa carpeta son *cdrom, sda, tty1* ou *usb*.

- */home:* cada usuario rexistrado no sistema terá unha carpeta */home/nome* da súa propiedade, e será nela na que almacena toda a súa información.

- */root:* esta carpeta será o *home* do administrador do sistema ou superusuario *root*.

- */srv: srv* é abreviatura de *service* e contén datos para os servicios activos no ordenador. Por exemplo, se temos un servidor web é normal que exista unha subcarpeta */srv/www*.

- */proc:* trátese dunha carpeta virtual, que xera o ordenador ao arrancar e está en continua modificación pois contén información dos procesos que se están executando no mesmo. Cada proceso en execución ten unha carpeta asociada, identificada polo seu PID. Outras carpetas que veremos serán *cpuinfo, devices* ou *uptime*.

- */sys:* outra carpeta virtual, tamén con información do sistema, pero neste caso non directamente asociada a procesos concretos, senón ao *kernel* do sistema e outros aspectos mais globais. Exemplos de arquivos habituais son *bus, devices* ou *kernel*.

- */run:* carpeta na cal moitas aplicacións do sistema almacenan información temporal. Exemplos habituais son *alsa, cups, mount* ou *udev*

- */var:* var é abreviatura de variable e refírese a información que está en continuo cambio, aínda que outras carpetas almacenen temén información variable (por exemplo /proc). Seguramente a subcarpeta mais coñecida é */var/log* que será onde se almacene a información de *log* xerada polo sistema e as distintas aplicacións (*apt, auth, kern, etc*).

- */media:* cando mais adiante falemos de *montar* dispositivos veremos que é preciso indicar un *punto de montaxe*. Pois esta carpeta será punto de montaxe habitual para medios extraibles, como poderia ser un CD ou un USB.

- */mnt:* de igual modo que no caso anterior esta carpeta emprégase como punto de montaxe pero para montaxes temporais, feitas manualmente polo usuario.

- */tmp:* contén arquivos temporais, habitualmente xerados pola distintas aplicacións, coa intención de non conservalos permanentemente. O usuario tamén pode colocar nel os seus arquivos temporais.

2.3. Acceso aos dispositivos

Ata agora non falamos para nada dos aspectos físicos do almacenamento, pero é evidente que cando abrimos un arquivo de texto para editalo ese arquivo estará en algún disco e en alguns sectores concretos de dito disco.

Para explicar como xestiona Linux o acceso aos dispositivos de almacenamento precisamos introducir varios aspectos, Imos empezar por analizar o contido da carpeta */dev* [6]. Se executamos o comando *ls -l /dev* obteriamos algo como o seguinte:

```
...
crw-rw-rw-  1 root  root        1,     8 oct 15 08:06 random
crw-------  1 root  root      249,     0 oct 15 08:06 rtc0
brw-rw----  1 root  disk        8,     0 oct 15 08:06 sda
brw-rw----  1 root  disk        8,     1 oct 15 08:06 sda1
brw-rw----  1 root  disk        8,     2 oct 15 08:06 sda2
brw-rw----  1 root  disk        8,     3 oct 15 08:06 sda3
...
```

Como vemos este listado é un pouco especial. Vemos que hai liñas que empezan con unha *b* e outras que empezan con unha *c*. As que empezan con *b* correspondense con dispositivos de bloque, que empregan o bloque como unidade de transferencia de información. As que empezan con *c* son dispositivos de tipo carácter, que transfiren a información *byte* a *byte*. O último campo é o nome do dispositivo ou *device*.

Fixémonos agora nos números que aparecen no quinto e sexto campo. O primeiro deles é o *major code* e correspondese co tipo de dispositivo concreto, e logo veremos que se traduce nun *device driver* específico. Por exemplo o *major code 8* correspóndese co primeiro disco SCSI conectado ao sistema, e o *1* corresponse cos dispositivos de tipo *memory* ou sexa, que non son dispositivos reais, senón *pseudodispositivos* [7] creados na memoria do ordenador.

O segundo campo denomínase *minor code* e define distintas *instancias* ou se queremos subtipos do mesmo dispositivo. Por exemplo co *major code* 1 veremos que hai distintos dispositivos, pero o denominado *random* ten o *minor code 8*. Este dispositivo é un xerador de números aleatorios. Co *major code* 8 vemos que

[6] https://www.linuxjournal.com/article/2597
[7] https://en.wikipedia.org/wiki/Device_file

hai varios dispositivos, o *sda* que ten o *minor code* 0 e o resto con números 1, 2, etc. Retomaremos este exemplo cando falemos das *particións*.

O sistema operativo será capaz de empregar todos os dispositivos para os cales teña asignado un *major code* e polo tanto dispoña do correspondente *device driver*. Cando nos ocupemos do *kernel* do sistema analizaremos onde están estes *drivers* e como se poden implementar uns novos. O listado de tipos de dispositivos configurados no sistema, cos seus correspondentes *major codes*, podemos consultalos no arquivo */proc/devices* [8].

Lembrar que o sistema operativo, ao detectar novo hardware conectado, se sabe como manexalo, xera automáticamente a correspondente entrada na carpeta */dev*. Por exemplo, se conectásemos un segundo disco SCSI poderiamos ver que se xerou unha nova entrada co *major code* 16 e *minor code* 0 e co nome *sdb*.

2.4. Particionado dos discos

Dende un punto de vista físico un disco duro non é mais que un conxunto de discos, cunhas cabezas lectoras/escritoras en cada un deles susceptibles de conter información binaria. Para poder empregalo como soporte no que almacenar información o primeiro paso é darlle formato. Para iso nos discos crease unha estructura lóxica que os divide en *pistas* e *sectores* [9]. As pistas son hipotéticos circulos concéntricos sobre os que so colocarán as cabezas lectoras para ler ou escribir datos. Cada pista dividirase a súa vez nunha serie de sectores cunha capacidade determinada de *bytes*, por exemplo 512 bytes ou 1 Kbyte. Polo tanto, a capacidade total do disco duro virá dada por: número de discos x número de pistas por disco x número de sectores por pista x capacidade do sector. Outros tipos de dispostivos (DVDs, memorias USB, etc) terán outra estructura diferente.

Actualmente as capacidades dos discos duros son moi elevadas e conleva certas vantaxes poder dividir dun xeito lóxico os discos para facer crer ao sistema operativo que en realidade temos varios discos, que se poden xestionar independentemente. Esto consiguese mediante o *particionado* do disco.

[8] https://www.oreilly.com/library/view/linux-device-drivers/0596000081/ch03s02.html

[9] https://es.m.wikipedia.org/wiki/Formato_de_disco

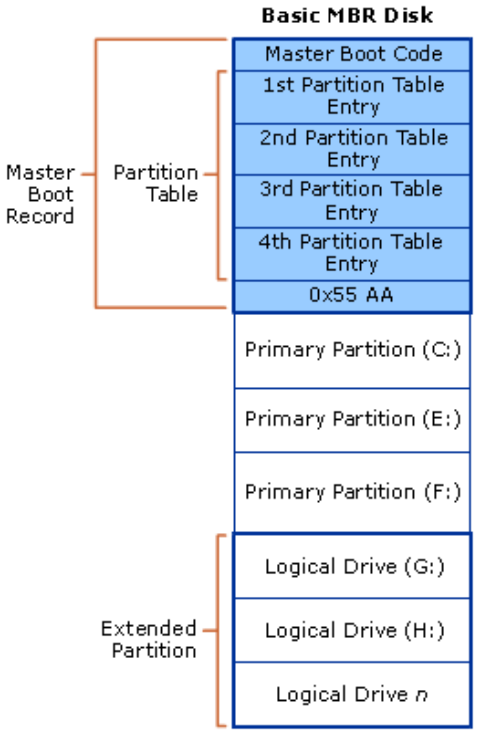

Figura 2.3: Particionado de disco segundo o sistema BIOS-MBR (sacado de [10])

2.4.1. BIOS-MBR fronte a UEFI-GPT

Ata non fai moitos anos o arranque dos ordenadores estaba dirixido pola *BIOS* do sistema. Esta é unha memoria ROM que contiña o código preciso para arrancar e chequear o hardware do sistema e despois comprobaba os distintos medios de almacenamento para comprobar se neles habia algún sistema operativo listo para arrancar. Para iso examinaba o primeiro sector do disco, un sector moi especial denominado *MBR* ou *master boot record*. De haber un sistema operativo instalado no disco nese sector estaría o código necesario para cargalo na memoria do ordenador e continuar o proceso de arranque. Pero tamén nese sector había espacio para conter a denominada *táboa de particións* (figura 2.3). O problema é que so había espazo para 4 particións, que se denominaban *particións primarias*.

Para posibilitar o uso de mais particións permitiuse que unha das particións, fose etiquetada como *partición extendida*. Nese caso, dentro dela poderian definirse mais particións, que se denominarían *particións lóxicas*, e que o sistema operativo podería manexar do mesmo xeito que as primarias [11].

[10] https://www.installsetupconfig.com/win32programming/windowsdiskapis2_2.html
[11] https://es.wikipedia.org/wiki/Partici%C3%B3n_de_disco

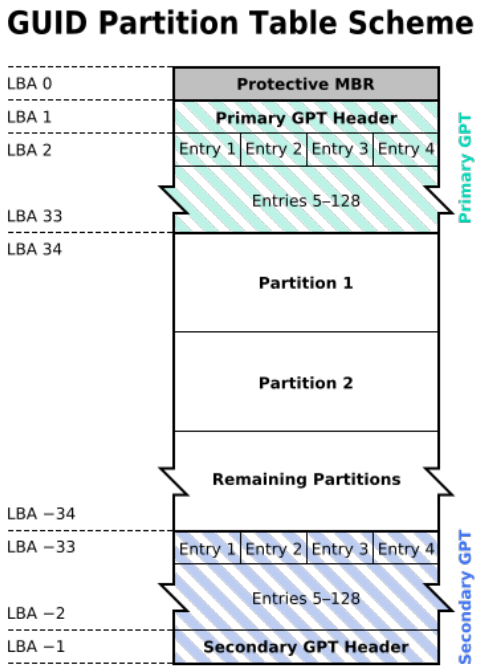

Figura 2.4: Particionado de disco segundo o sistema UEFI-GPT (sacado de [12])

Nos ordenadores mais moderos a BIOS foi sustituida por un novo estandar denominado *UEFI* ou *unified extensible software interface*. Aínda que este novo estándar permitiría empregar a MBR admite un novo medio de formato de disco denominado GPT (*GUID partition table*).

Tal como se ve na figura 2.4 agora o sector 0 do disco non se emprega e a táboa de particións colocase nos sectores 1-33, podendo conter ata 128 particións. Ao final do disco almacénase unha copia de seguridade da táboa.

Na parte práctica veremos os comandos que permiten crear ou eliminar particións nos discos.

2.4.2. Particións típicas

Vexamos agora as particións mais habituais en sistemas Linux.

- *Raíz (/)*: esta é a única partición obrigatoria. En caso de que sexa única conterá toda a árbores do sistema de arquivos, todas as carpetas e todos os arquivos do sistema. O tamaño mínimo recomendable serían uns 10 GB,

[12] https://es.wikipedia.org/wiki/Tabla_de_particiones_GUID

pero se é a única partición e inclue polo tanto os arquivos de usuario mellor que teña polo menos 20 GB.

- *swap*: a partición de tipo *swap* sería a única que non formaría parte do sistema de arquivos. Sería un espazo en disco habilitado para ampliar a memoria principal (RAM) do sistema (a este esquema denomínaselle memorial virtual), tanto para poder executar mais programas a vez, como para permitir a *hibernación* do sistema. Antes recomendábase crear un espazo de *swap* dun tamaño dobre ao da RAM, pero cos tamañao actuais de RAM soense empregar áreas de *swap* menores.

- */boot* aínda que non é obrigatoria recoméndase crear esta partición para aloxar o *kernel* de Linux. Soe ser unha partición pequena, duns 250 MB.

- */home*: presente en casi todos os equipos está destinada a aloxar os arquivos propios dos usuarios normais do sistema, e separalos dos arquivos propios do sistema operativo. Unha das vantaxes deste esquema é que, en caso de reinstalación do sistema, os arquivos do usuario non se tocan. Tamén simplifica as copias de seguridade.

- */boot/efi*: nos sistemas que seguen o estándar UEFI-GPT é obrigatoria a existencia desta partición. Nela aloxariase o código de arranque dos diferentes sistemas operativos presentes no sistema. Soen ter un tamaño pequeno, de 250 MB aproximadamente.

Volvendo ao símil da biblioteca, poderiamos comparar os edificios con discos. Neles poderiamos definir habitacións, que serían as particións. Pero de momento témolas valeiras, preparadas para conter calquera cousa.

2.5. Creación do sistema de arquivos

Se queremos almacenar libros nunha habitación o primeiro que haberá que facer é crear unha estructura axeitada para almacenalos e operar con eles: buscar un libro dado o seu código, incorporar novos libros, etc. No caso da biblioteca farémolo colocando andeles e definindo unha organización que permita localizar facilmente os libros.

[12] https://www.atareao.es/como/particiones-en-linux/

Figura 2.5: Estrutura da táboa de inodos (sacado de [14])

Volvendo ao noso caso, para que unha partición poda conter arquivos e carpetas é preciso darlle un formato axeitado. Trátase de organizar o espazo desa partición en bloques e crear un índice que nos indique, para cada un dos arquivos almacenados, en que sector ou sectores está almacenado. Tamén deberá de marcar de algun xeito os bloques libres para saber onde deberemos gardar os novos arquivos. Cando borremos un arquivo este deberá desaparacer da táboa de índices e os bloques que ocupaban serán marcados como libres.

A esta estructura denomínaselle "sistema de arquivos"(*filesystem*). Cómo se verá na parte práctica o comando para crear dita estrutura nunha partición é *mkfs*. A este proceso soese denominar *formateo de alto nivel* [13].

2.5.1. Táboade de inodos

Ao instalar un sistema de arquivos nunha partición o que facemos é crear ao inicio da mesma unha *táboa de inodos*, cunha estrutura semellante a amosada na figura 2.5.

Dita táboa contén en primeiro lugar un *bloque de carga*, do que non imos a falar porque non nos afecta, e logo un *superbloque*. Este superbloque é o que contén información sobre o propio sistema de arquivos. No seguinte listado amośase parte do que se obtén co comando *dumpe2fs* nunha partición concreta.

```
Last mounted on:          /home
Filesystem UUID:          9d0a117f-01bf-4d4c-9902-0d77fa9f82b2
Filesystem features:      has_journal ext_attr resize_inode
Filesystem OS type:       Linux
Inode count:              12214272
```

[13] https://es.m.wikipedia.org/wiki/Formato_de_disco
[14] http://mural.uv.es/oshuso/821_caractersticas_del_sistema_de_ficheros_de_linux.html

```
Block count:              48828160
Free blocks:              40693728
Free inodes:              12049383
Block size:               4096
```

Agora non imos explicar o significado desas liñas, pero vemos que aparece duas veces o nome *inode* ou inodo en castelán. Un inodo é unha estructura de datos pensada para almacenar toda a información dun arquivo en Linux [15]. Todos os arquivos en Linux están identificados polo seu número de inodo, o seu nome non é mais que un atributo do mesmo.

A terceira sección contén precisamente a lista de inodos, un por cada arquivo presente no sistema. Se nos fixamos no listado anterior, o campo *Inode count* estanos decindo precisamente que esa partición ten espacio para conter unha lista de ata 12.214.272 inodos, e polo tanto o mesmo número de arquivos. Este número defínese no momento da creación do sistema de arquivos, ben por defecto ou cunha opción específica (*mkfs -N numero-de-inodos*). Se nos fixamos agora no campo *free inodes* vemos que nesa partición hai aínda 12.049.383 inodos libres.

Cada inodo ocupa habitualmente 256 bytes e contén información como o identificado do dispositivo no que está o arquivo, a lonxitude do arquivo o identificador do propietario, a máscara de permisos, marcas de tempo (data de creación, acceso e última modificación), etc.

O resto do espazo da partición será o espazo realmente libre para conter os arquivos. Ese espazo estará divido en bloques, que será a unidade mínima de almacenamento. Un arquivo poderá ocupar un ou varios bloques, contiguos ou non.

2.5.2. Tipos de sistemas de arquivos

Linux soporta unha gran variedade de tipos de sistemas de arquivos, tanto por compatibilidade con outros sistemas operativos (como Windows ou MacOs) como para adaptarse a diferentes situacións (por exemplo xestión óptima de moitos arquivos pequenos). O sistema de arquivo por defecto en Linux denomínase *ext4*, que é unha evolución do *ext3*, que á sua vez é unha evolución do *ext2* [16]. Cada un deles vai mellorando características do anterior, como o tamaño máximo dos

[15] https://es.wikipedia.org/wiki/Inodo
[16] http://mural.uv.es/oshuso/822_tipos_de_sistemas_de_ficheros_en_linux.html

arquivos, a incorporación do *journaling* [17] ou a eficiencia de lectura/escritura. Outros sistemas de arquivos válidos son ReiserFs, JFS ou XFS. Linux permite tamén manexar particións de tipo FAT ou NTFS, propias de Windows, HFS de MacOS ou tamén formatos como NFS ou SMB, característicos de sistemas accesible en remoto.

2.6. Montaxe

Voltemos ao simil da biblioteca para explicar este último punto. Nos queremos ter unha *biblioteca virtual única*, de xeito que os usuarios non teñan por qué coñecer os detalles da organización interna. Eles únicamente buscan un libro nela, solicítano ou devolveno. Pero tamén queremos que a biblioteca poda ocupar un único *edificio principal* ou poda extenderse a outros edificios. Por exemplo a sala dos libros de "historia de Roma"poderian ocupar unha habitación do edificio principal ou ben, extenderse ata un edifico anexo. Nese caso precisamos de algun sistema que *enlace* ese edificio co principal, para que este o perciba como unha habitación mais. Dende un punto de vista funcional sería algo asi como se nesa habitación concreta houbese un sistema de "correo"propio que trasladase as peticións que se reciban ata o edificio anexo de xeito transparente.

Nos queremos ter un sistema de arquivo virtual único, cunha estrutura xerárquica que empece no directorio raíz e abarque todo o sistema. Deste maneira calquera arquivo accesible terá un *path* absoluto único, independentemente de onde resida físicamente. O mecanismo que nos permite facer esta ponte denominase montaxe (*mount*).

Na figura 2.6 vemos un exemplo concreto. O sistema raíz reside no dispositivo */dev/hda1*, que terá un formato axeitado, por exemplo un *ext4*. Temos un cdrom insertado na correspondente unidade formateado con outro tipo de *filesystem*, por exemplo ISO9660. Para poder acceder a el é preciso montalo.

Para montar un dispositivo o primeiro que se necesita é un *punto de montaxe*, unha carpeta valeira na que o sistema crerá que está toda a información do dispositivo. No noso caso esa carpeta será */mnt/cdrom*. A continuación poderiamoms executar o seguinte comando:

[17] https://es.wikipedia.org/wiki/Journaling

[18] http://etutorials.org/Linux+systems/red+hat+linux+9+professional+secrets/Part+II+Exploring+Red+Hat+Linux/Chapter+12+Basic+System+Administration/Managing+the+File+System/

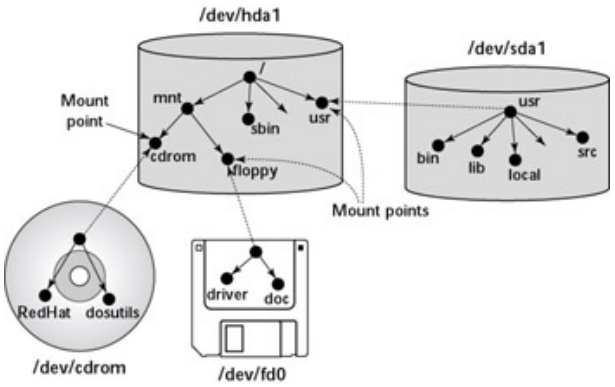

Figura 2.6: Exemplo de montaxe en Linux (sacado de [18])

```
mount -t iso9660 /dev/cdrom /media/cdrom
```

Ese comando indica ao sistema que o dispositivo */dev/cdrom* contén información estruturada de acordo ao estándar iso9660, e que nos queremos que esa información sexa accesible como si estivese na carpeta */media/cdrom*.

A partires dese momento, se executamos o comando *ls/media/cdrom* veremos exactamente o contido do cdrom e poderemos operar con el sin importarnos onde está físicamente.

Na mesma figura vemos como a carpeta */usr* está no dispoitivo */dev/sda1* ou que tamén temos un disquete que foi montado na carpeta */mnt/floppy*.

2.6.1. O arquivo */etc/fstab*

No proceso de arranque do sistema realizanse de xeito automático certos procesos de montaxe. Por exemplo, se as carpetas dos usuarios están nun dispositivo propio, é preciso que ese dispositivo se monte na carpeta */home* antes de que estes accedan ao sistema. A maneira de facelo é a través dun arquivo que indica o modo de montaxe dos dispotivos coñecidos. Ese arquivo é o */etc/fstab*.

No seguinte listado amósase un exemplo concreto de arquivo */etc/fstab*.

```
# /etc/fstab: static file system information.
#
# <file system>     <mount point>   <type>  <options> <dump>  <pass>
# / was on /dev/sda6 during installation
UUID=8f778349-c1a1  /           ext4    errors=remount-ro   0   1
# /boot/efi was on /dev/sda1 during installation
```

```
UUID=0656-E5E1      /boot/efi   vfat    umask=0077          0   1
# /home was on /dev/sda7 during installation
UUID=9d0a117f-01bf /home        ext4    defaults            0   2
# swap was on /dev/sda8 during installation
UUID=4209d835-f495 none         swap    sw                  0   0
```

As liñas que empezan por # son comentarios explicativos, polo tanto unicamente temos 4 liñas que analizar. A primeira indica que o dispositivo con UUID 8f778349-c1a1, formateado en *ext4*, contén a raíz do sistema de arquivos. Non imos explicar aquí o resto de campos [19]. Actualmente téndese a identificar os dispositivos polo seu *identificador único* ou UUID, un número creado durante o proceso de creación da partición. O comentario previo acláranos que en realidade é o dispositivo */dev/sda6*.

A segunda liña indica cal é a partición EFI, aquela a que se accede durante o proceso de arranque. Neste caso vese que está en */dev/sda1* e que será montada en */boot/efi*. Fixarse que esta partición está formateada en *vfat*.

A terceira liña indica que na partición */dev/sda7* está a carpeta */home*. E a cuarta liña indica que a área de swap está na partición */dev/sda8*.

2.7. Volumes lóxicos

Ata aqui o funcionamento do sistema de arquivos en Linux ata fai ... 5 anos. Pero este sistema tiña un problema importante de *flexibilidade*. Que pasa se a partición */home* se nos queda corta? Cómo poderiamos ampliala? Non hai unha solución fácil, aínda que hoxe en día hai ferramentas para ampliar ou reducir o tamaño das particións. Pero que pasa se non temos mais espazo no disco e mercamos outro, podemos ampliar a partición para que colla espazo no segundo disco? Aquí si que a resposta é non.

Para solucionar estes é outros problemas apareceu o concepto de *volumes lóxicos*. O que se fai é meter unha capa de software intermedia entre os dispositivos físicos (discos e particións) e os sistemas de arquivos. A esta capa denomínaselle *logical volume manager* ou *LVM* [20].

O LVM manexa tres conceptos fundamentais:

[19] https://www.howtogeek.com/howto/38125/
[20] https://blog.inittab.org/administracion-sistemas/lvm-para-torpes-i/

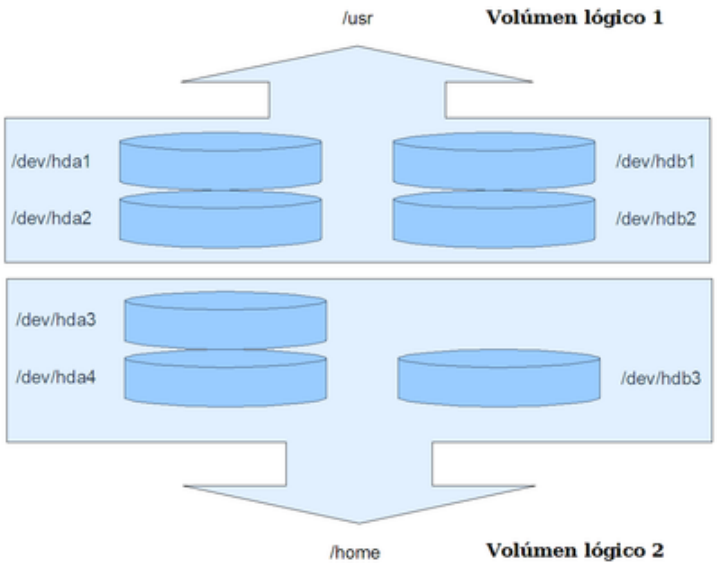

Figura 2.7: Esquema básico de LVM (sacado de [21])

- *Volume físico*. Serían as particións físicas dos diferentes discos. Para que una partición física sexa manexada polo LVM existe un comando (*pvcreate*) para convertir a partición nun volume físico.

- *Grupo de volumes*. Este é o elemento innovador e esencial. Podemos velo como unha caixa a que se poden engadir volumes fisicos, independentemente de se están nun disco ou en outro, para construir un *espazo unificado de almacenamento*. Outra vantaxe é que un *grupo de volumes* pode crecer ou decrecer dinamicamente, ou sexa que podemos engadir novos volumes físicos ou quitalos en calquera momento.

- *Volume lóxico*. Sería o equivalente a unha partición formateada cun sistema de arquivos. Ou sexa, trátase dun espazo no cal se poden aloxar arquivos. A novidade é que agora estes volumes lóxicos creanse nos *grupo de volumes*, e polo tanto será transparanente para o usuario a que dispositivos físicos (*volumes físicos*) corresponde.

Para explicalo imos recorrir ao esquema amosado na figura 2.7.

Como vemos temos dous discos diferentes *sda* e *sdb*. O primeiro ten 4 particións e o segundo 3. O primeiro que facemos é crear 7 *volumes físicos*, un por partición. A continuación, se queremos manexar os dous discos en conxunto, facemos un

[21] https://es.wikipedia.org/wiki/Logical_Volume_Manager

grupo de volumes que englobe a todos os *volumes físicos* (particións). E por último creamos 2 *volumes lóxicos* indicando explícitamente que *volumes físicos* queremos asignar a cada un deles. Se non nos importa en que particións concretas estén esas carpetas podemos crear os *volumes lóxicos* indicando soamente o tamaño. Nese caso é o LVM o que deciden onde almacenalas.

A vantaxe deste sistema é que se no futuro precisamos, por exemplo, aumentar o espazo da carpeta */home* podemos facelo sin mais que asigarlle mais espazo libre dentro do *grupo de volumes*. E se non queda espazo libre podemos ampliar o propio *grupo de volumes* con novos discos ou particións.