Programa de Administración de Sistemas e Redes

Tomás Fernández Pena

9 de setembro do 2014

Administración de Sistemas e Redes - [G4011322]

Programa

1. Introducción
   - 1.1. Introducción á materia
     - A figura do administración de sistemas
     - Obxectivos da materia
     - Por que UNIX/Linux?
     - Información oficial
     - Relación con outras materias.
   - 1.2. Tarefas dun administrador de sistemas
     - Principais tarefas
     - Fontes de información para o administrador
   - 1.3. Políticas e estándares
     - Políticas e procedementos
     - Estándares e recomendacións
     - Lecturas adicionais
2. Introdución aos sistemas Linux/Unix
   - 2.1. Introdución a Unix y Linux
     - Historia de Unix
     - Sistemas GNU/Linux
   - 2.2. Instalación do sistema e de software
     - Tipos de servizos
     - Virtualización
     - Instalación de Linux Debian
     - Verificación da instalación
     - Instalación de software
   - 2.3. Uso da liña de comandos
     - O interprete de comandos (shell)

- A liña de comandos
- Comandos básicos
- Variables de shell
- Expansións da shell
- Redirección da entrada/saída
- Orden de avaliación
- Ficheiros de inicialización de bash

2.4. Programación de scripts de administración

- Programación shell-script

2.5. Manexo de ficheiros de texto

- Expresións regulares
- Comandos para o procesamento de textos

2.6. Programación en Python

- Introdución a Python
- Tipos de datos en Python
- Control de fluxo
- Orientación a obxetos
- Procesamento de textos
- Outros aspectos
- Subprocesos
- Outros módulos de interese
- Exemplos

2.7. Introdución a Perl e Ruby

- Perl
- Ruby

3. Actividades administrativas básicas

3.1. Comandos básicos para a xestión de procesos

- Ver os procesos en execución
- Manipulación de procesos
- Manexo da prioridade e recursos dun proceso
- Análise básica do rendemento do sistema
- Ferramentas gráficas
- O directorio /proc

3.2. Xestión do sistema de ficheiros

- Tipos de ficheiros e atributos
- Enlaces
- Localización de ficheiros
- Particións e sistemas de ficheiros
- Sistemas de ficheiros con LVM
- Manexo de discos cifrados

3.3. Xestión de usuarios

- Ficheiros de información dos usuarios
- Creación manual dunha conta
- Comandos para a xestión de contas
- Cuotas de disco

3.4. Instalación e configuración básica de redes de área local
- Comandos de configuración da rede
- Ficheiros de configuración da rede
- Configuración do DHCP

3.5. Automatización de tarefas
- Tarefas periódicas
- Automatización da configuración

3.6. Copias de seguridade
- Estratexias para as copias de seguridade
- Comandos básicos
- Outras aplicacións

4. Servizos básicos de servidor a cliente

4.1. Introdución

4.2. Acceso remoto e transferencia de ficheiros
- Servicio de telnet y ftp
- SSH

4.3. Sistemas de ficheiros en rede
- Características principais
- Instalación de NFS en Debian
- Consideracións de seguridade en NFS

4.4. Servizos de directorio
- Servizo de Información de Rede (NIS)

4.5. Servizo de directorio: LDAP
- OpenLDAP
- Modelo de datos de LDAP
- Instalación dun servidor LDAP
- Migración desde ficheiros ou NIS
- Instalación dun cliente LDAP
- Configuración de LDAP con múltiples servidores
- Ferramentas de administración de LDAP

Bibliografía

Libro base

[1] *UNIX and Linux System Administration handbook*, Evi Nemeth, Garth Snyder, Trent R. Hein e Ben Whaleyr, Prentice Hall, 2010. Libro base da materia. Última edición dun clásico, incluíndo información sobre varias distribucións Linux (Ubuntu, RedHat e openSuse), así como outros sistemas Unix (Solaris, HP-UX e AIX).

Libros complementarios

[2] *Essential System Administration* (3ª ed.), Æleen Frisch,O'Reilly, 2002. Libro da serie de O'Reilly para a administración Unix.

[3] *LPIC 1 Certification Bible*, Angie Nash, Jason Nash, Hungry Minds, 2001. Orientado á preparación do exame para a obtención do certificado 1 do Linux Professional Institute (www.lpi.org/eng/certification/the_lpic_program/lpic_1).

[4] *Manual de Administración de Linux*, Steve Shah, MC Graw Hill, 2001. En castelán.

[5] *Python for Unix and Linux System Administration*, Jeremy M. Jones e Noah Gift, O'Reilly, 2008. Libro sobre programación en Python para administradores de sistemas.

Documentación en Internet

[6] *The Debian Administrator's Handbook*, Raphaël Hertzog e Roland Mas, 2012, debian-handbook.info/browse/stable/

[7] *Administración Avanzada de GNU/Linux*, Josep Jorba Esteve e Remo Suppi Bolditro, UOC Formación de Posgrao, Marzo 2004. www.uoc.edu/masters/softwarelibre/esp/materials/Admin_GNULinux.pdf. Libro para un curso de formación de posgrao da Universidade Oberta de Catalunya (UOC).

[8] *The Linux System Administrators' Guide*, Lars Wirzenius, Joanna Oja, Stephen Stafford e Alex Weeksk, xullo 2005. tldp.org/LDP/sag. En *The Linux Documentation Project* (tldp.org) podemos atopar guías, HOWTOS e FAQs sobre Linux en xeral e a súa administración en particular.

[9] *The Linux Network Administrator's Guide, Second Edition*, Olaf Kirch e Terry Dawson, marzo 2000. www.tldp.org/LDP/nag2.

[10] *Linux System Administration Made Easy*, Steve Frampton, novembro 1999. www.tldp.org/LDP/lame/LAME/linux-admin-made-easy.

[11] *Advanced Bash-Scripting Guide*, Mendel Cooper, última revisión marzo 2009, www.tldp.org/abs/html. Titorial e referencia de programación script con Bash.

Tema 1: Introducción a la administración de sistemas

Administración de Sistemas e Redes

Tomás Fdez. Pena

tf.pena@usc.es

Índice

1. Introducción a la asignatura

1.1. La figura del administrador de sistemas

- El administrador de sistemas es quien tiene la capacidad y la responsabilidad de establecer las acciones, procedimientos y normas para conseguir:
  - asegurar que el sistema funcione correcta y eficientemente, y

    - asegurar que todos los usuarios pueden usar el sistema de manera fácil y sin problemas
- Tarea esencial, dado el incremento en la complejidad de los sistemas y redes

1.2. Objetivos de la asignatura

- Adquirir competencias de un Administrador de Sistemas a nivel intermedio
    - Facilidad de uso de la mayor parte de los aspectos de la administración de sistemas GNU Linux/UNIX
    - Conocimiento de administración de redes
    - Capacidad de entender y escribir scripts de administración
    - Capacidad de identificar tareas que requieran automatización y automatizarlas
    - Capacidad de resolver problemas rápida y completamente

1.3. ¿Por qué UNIX/GNU Linux?

- UNIX tiene una larga historia en la industria y la educación
- UNIX/GNU Linux es popular a nivel de empresa
- Es independiente del hardware
- GNU Linux es abierto y gratuito
- GNU Linux proporciona prácticamente todo el software necesario para un sistema completo

¿Por qué no Windows?

- Dependencia de una sola empresa
- Es caro
- No es completo
- Es cerrado
- Oculta su complejidad
- Conociendo Linux es fácil aprender otros SO

1.4. Información oficial

- 6 ECTS $\longrightarrow$ 5 horas expositivas/45 laboratorio
- Asistencia obligatoria a clases de laboratorio
- Evaluación: 60 % examen teoría/40 % evaluación prácticas
- Apuntes disponibles en el CV o en www.ac.usc.es/docencia/ASR/
- Profesorado:
  - Teoría: Tomás Fernández Pena
  - Prácticas: Francisco Argüello Pedreira

Información completa:
www.usc.es/gl/centros/etse/materia.html?materia=85056&ano=65

1.5. Relación con otras asignaturas

Administración Avanzada de Sistemas e Redes. Optativa, trata aspectos de monitorización y optimización de servidores, virtualización, gestión de redes y administración de servicios (web, e-mail, etc.).

Seguridade Informática. Obligatoria, cubre aspectos de seguridad en redes (cortafuegos, VPNs, IDs, etc.).

Enxeñaría de Computadores Obligatoria, instalación y configuración de centros de procesado de datos e instalaciones informáticas de tamaño medio/grande (servidores, virtualización y consolidación, redes de almacenamiento, backups, etc.).

Administración de Bases de Datos. Optativa, trata la administración de sistemas de gestión de bases de datos (instalación y gestión, seguridad, backups, etc.).

2. Tareas de un administrador de sistemas

Tareas comunes:

- Administrar el hardware de los sistemas
- Administrar las aplicaciones instaladas

- Administrar usuarios
- Comprobar el buen funcionamiento del sistema
- Contabilizar el uso de recursos por parte de los usuarios
- Administración de la seguridad
- Mantenimiento de la documentación
- Soporte técnico a usuarios

Tareas específicas dependen del entorno donde desarrolle su trabajo el administrador, incluyendo tipos de usuarios, hardware/software disponible, políticas de gestión, etc.

1. **Entorno de trabajo**
   - Entorno de trabajo (empresa, administración pública, etc.) compuesto de :
     - usuarios (de 10 a 1000s),
     - recursos materiales (posiblemente operando 24x7),
     - información (software, archivos, etc.)
     - políticas de gestión
2. **Usuarios**
   - Algunas características:
     - número,
     - experiencia (o no) en el uso de sistemas informáticos,
     - tipo de trabajo,
     - responsabilidad o irresponsabilidad
3. **Hardware/Software**
   - Los ordenadores, software, redes, impresoras, etc. influyen en el tipo de trabajo del administrador
     - número y complejidad,
     - una o varias redes,
     - sistemas homogéneos o heterogéneos, corriendo el mismo o diferente SO y software de aplicaciones:
       - PCs tradicionales

        - servidores sin monitor
        - sistemas sin interfaz gráfico
        - sistemas sin disco
        - sistemas con varias CPUs (SMP, clusters, etc.)
        - ...

4. **En cualquier caso**

    - El administrador es un instrumento para facilitar la vida a los usuarios, no para complicársela:
        - Tener siempre presente el espíritu de servicio
        - Informar a los usuarios de los cambios en el sistema que les afecten, personalmente o mediante herramientas de comunicación, sin llegar a saturar
        - Ayudar en los problemas que surjan a los usuarios
        - Atender sus quejas con educación, pero ser capaz de imponer su autoridad cuando sea preciso
        - No olvidar que los usuarios no son expertos y su conocimiento informático puede ser nulo

2.1. Principales tareas

Cuatro categorías:

- operaciones diarias
- hardware y software
- interacción con los usuarios
- organización y planificación

1. **Operaciones diarias**

    - Tareas que deben realizarse regularmente, por ejemplo:
        - Añadir y borrar usuarios
        - Monitorizar el sistema:
            - uso de recursos: CPU, memoria, discos, etc.
            - actividades irregulares de los usuarios
            - problemas inesperados
        - Realizar copias de seguridad

- Muchas de esas tareas pueden (y deberían) ser automatizadas
  - se evita *reinventar la rueda*
  - simplifica el trabajo
  - permite delegar en otros

2. Hardware y software

- Algunas tareas relacionadas con hardware y software:
  - evaluación y compra,
  - instalación y mantenimiento,
  - prevención de problemas,
  - actualización de hardware y software,
  - eliminación y migración de sistemas antiguos

3. Organización y planificación

- El administrador debe ser capaz de anticiparse a los problemas (proactividad)
  - necesidad de organización y planificación
- Elementos a tener en cuenta:
  - documentación: para el administrador, los usuarios y la dirección
  - gestión del tiempo de trabajo
  - planificación a medio y largo plazo
  - actualización de conocimientos
  - automatización de actividades repetitivas

4. Documentación

- Posiblemente la tarea más aburrida, pero una de las más importantes,
  - documentación acerca de los detalles de cada sistema particular:
    - localización, detalles de compra, ...
    - software instalado, usuarios, ...
    - ficheros de configuración, ...
  - etiquetado del hardware
    - información sobre el nombre, IP, MAC, etc. de cada sistema particular
  - libro de cambios en los sistemas
    - modificaciones en los ficheros de configuración, actualizaciones, ...

- informes sobre las tareas realizadas
  - facilitan resolver un problema cuando ocurre por segunda vez
- documentación para usuarios sobre uso de los sistemas

2.2. Fuentes de información para el administrador

Una parte esencial de la tarea del AS es diagnosticar y resolver problemas

Para resolver un problema debemos:

1. mantener la calma
2. consultar las fuentes de información disponibles:
   - conocer los sitios donde mirar,
   - antes de preguntar, intentar ver si el problema ya está descrito,
   - al preguntar, ser educado, conciso e informativo
3. registrar la solución
4. contribuir a la comunidad

No podemos conocer todo lo necesario para resolver todos los problemas, pero si debemos saber donde acudir en busca de información.

Fuentes de información:

- Asociaciones profesionales y grupos de usuarios:
  - Galicia: OSL-CIXUG, GPUL, Ghandalf, GALPon, AGNIX, etc.
  - España: Hispalinux, BULMA, etc.
  - Internacional: USENIX, LISA, ACM, IEEE Computer, etc.
- Libros y revistas:
  - En general, referenciados por el nombre RTFM (Read The F. Manual)
  - Multitud de libros y revistas sobre UNIX/Linux, algunas online como LinuxGazette, LinuxFocus o OpenSource Subnet
  - A destacar la colección de O'Reilly
- Internet
  - Es la principal fuente de información para administradores y usuarios

Fuentes en Internet

En Internet podemos encontrar:

- software,
- información (documentación, guias, HOWTOS, etc.)
- foros de discusión

Software

- paquetes RPM: rpmfind.net
- paquetes Debian: www.debian.org/distrib/packages
- paquetes en código fuente: sourceforge.net
- ...

Documentación

- The Linux Documentation Project (www.tldp.org) mantiene guías, FAQs, HOWTOs

  HOWTO Documentos sobre aspectos específicos de Linux, por ejemplo:
  - *Partition HOWTO*, *NFS-HOWTO*, *Network Install HOWTO*, etc.
- Howtoforge (www.howtoforge.com) tutoriales varios
- Documentación de empresas y distribuciones, por ejemplo:
  - IBM developerWorks, por ejemplo LPI[1] certification 101/2 o LPI certification 201/2
  - Documentación de Red Hat (www.redhat.com/docs/)
  - Documentación de Debian (www.debian.org/doc/index.en.html)

Foros de discusión

- Grupos de noticias:
  - instalación y administración:

[1]LPI: Linux Professional Institute: organización sin animo de lucro que proporciona certificaciones para administradores Linux

        - comp.unix.admin, comp.os.linux.setup, es.comp.os.linux.instalacion,...
    - redes:
        - comp.os.linux.networking, es.comp.os.linux.redes,...
    - seguridad:
        - comp.security.unix, comp.os.linux.security,...
    - varios:
        - comp.unix.misc, comp.os.linux.misc, es.comp.os.linux.misc,...
    - para buscar algo concreto: groups.google.com
- Listas de correo:
    - existen listas de correo para multitud de tópicos:
        - listas de correo de Debian (www.debian.org/MailingLists)
        - listas de correo de GNOME (mail.gnome.org), o KDE (lists.kde.org)
        - ...
    - otros foros:
        - LinuxForums
        - LinuxQuestions
        - ...

Otras fuentes de información

- Compatibilidad hardware de Linux:
    - www.linuxhardware.net
    - Linux Hardware Compatibility HOWTO
    - Impresoras: OpenPrinting, USBs linux-usb.org, laptops, etc.
    - ...
- Trucos, ayuda,
    - BULMA
    - Linux Tips & Tricks
    - Linux Help
    - Just Linux
    - ...
- Seguridad
    - Kriptopolis: Información y noticias sobre criptografía y seguridad (en castellano)

    - CERT (de la Carnegie Mellow University): anuncios de seguridad, parches, etc.
    - SecurityFocus.com: especializado en noticias e información sobre seguridad
    - SANS: *the System Administration, Networking, and Security Institute*
    - ...
- Noticias, información, comentarios, enlaces, etc.
    - UGU: Unix Guru Universe; material para administradores UNIX
    - SlashDot: noticias para *nerds*
    - Freecode: información sobre aplicaciones *open source*
    - LinuxHQ: guías fáciles para usuarios de Linux
    - LinuxPlanet: discusiones, foros, tutoriales, etc.
    - Linux Today: actualidad Linux
    - Linux.org: promueve el uso de Linux
    - Linux.com: información para empresas
    - kernel.org: Linux Kernel Archives
    - sunHelp: una buena colección de enlaces a recursos para los administradores
    - ...

3. Políticas y estándares

Políticas de gestión

- Definen el qué, por qué y cómo hacer las cosas en la organización
    - determinan el tipo de uso que los empleados pueden hacer de los sistemas, p.e.
        - derechos de acceso a los recursos,
        - limites en el uso de recursos (disco, CPU, etc.),
        - ¿puede el AS acceder al e-mail de los usuarios?
- Normalmente, las políticas de gestión son responsabilidad de la gerencia o dirección técnica de la organización
    - el administrador debe respetar y hacer respetar esas políticas

    - no inventar políticas de uso de recursos que contradigan las políticas generales
  - La obediencia no debe ser ciega: si se considera que una política no se puede implantar o se puede mejorar, hay que dialogar con los superiores
  - Junto con las políticas, se debe establecer un sistema de penalizaciones

3.1. Políticas y procedimientos

Necesidad de documentos de políticas escritas y firmadas

- Políticas de los servicios administrativos
- Derechos y responsabilidades de los usuarios
- Políticas aplicables a los administradores de sistemas
- Políticas de usuarios temporales

Todos los usuarios deberían firmar un documento de conformidad

- Ejemplo en la página 947 del libro base

Ejemplos de políticas

- Normativa de la USC
- Ejemplo de la Universidad de California y el Centro de Supercomputación de San Diego

Seguridad

Toda política está relacionada con la estabilidad y seguridad del sistema

- debe quedar claro qué proteger y las prioridades

Necesidad de obtener un equilibrio

- Servicios ofrecidos vs. seguridad proporcionada (más servicios = menos seguridad)
- Facilidad y comodidad de uso vs. seguridad (seguridad=1/comodidad)
- Coste de la seguridad vs. riesgo (coste) de las pérdidas

Necesario crear guías de seguridad

- Ver ejemplo en el Centro de Supercomputación de San Diego

Necesaria una política de recuperación de desastres

- Ver libro base, sección 30.10

Procedimientos

Recetas para realizar determinadas tareas

- Añadir o dar de baja un host
- Añadir o dar de baja a un usuario
- Actualizar un sistema
- Realizar copias de seguridad de los sistemas
- Realizar apagados de seguridad
- etc.

Recetas por escrito y siempre disponibles

- Pueden tomar la forma de scripts comentados
- Un nuevo administrador debe ser capaz de entenderlas rápidamente

3.2. Estándares y recomendaciones

Existen estándares para la gestión correcta de las infraestructuras de IT y la seguridad

- Esquemas de certificación para valorar las instalaciones

Estándares relacionados con la seguridad

1. ISO 27002 (anteriormente ISO 17799, basado en el británico BS 7799-1)
    - Recomendaciones para realizar la gestión de la seguridad de la información dirigidas a los responsables de iniciar, implantar o mantener la seguridad de una organización
    - No es una norma tecnológica

        - proporciona buenas prácticas neutrales con respecto a la tecnología y a las soluciones disponibles en el mercado
    - Forma parte del grupo de estándares ISO 27000
    - Adaptación española: UNE-ISO/IEC 17799
    - Estándares relacionados:
        - ISO 27001, especifica requisitos para establecer, implementar, mantener y mejorar un sistema de gestión de la seguridad de la información consistente con ISO/IEC 27002, reemplaza al BS7799-2
        - UNE 71502:2004, basado en BS 7799-2

2. *Standard of Good Practice*
    - Documentación detallada que identifica buenas prácticas en seguridad de la información
    - Creado por el Information Security Forum

3. RFC 2196, *Site Security Handbook*
    - Documento práctico con recomendaciones sobre aspectos de seguridad, políticas de usuarios y procedimientos
    - Proporciona una visión general sobre la seguridad de la información, incluyendo seguridad de red, respuesta a incidentes o políticas de seguridad

Otros estándares

1. ISO/IEC 20000
    - Estándar internacional en gestión de servicios de Tecnologías de la Información
    - Basado en el estándar británico BS 15000, esarrollado en 2005 y revisado en 2011/12
    - Tiene varias partes, la más importante la ISO/IEC 20000-1:2011
        - define los requerimientos necesarios para garantizar una entrega de servicios de TI con calidad aceptable para todos los clientes (reemplaza 20000-1:2005)

2. ITIL (*Information Technology Infrastructure Library*)

- Conjunto de documentos de buenas prácticas para la gestión de servicios de TI
- Proporciona un marco de trabajo adaptable de buenas prácticas para ayudar a las organizaciones a lograr calidad y eficiencia en las operaciones de TI
- ITIL v3 (mayo 2007, actualizado en julio 2011) consta de 5 libros: *Service Strategy*, *Service Design*, *Service Transition*, *Service Operation* y *Continual Service Improvement*
- Cuatro niveles de certificación ITIL v3 para profesionales: Foundation, Intermediate, Expert y Master (no certificación para organizaciones)

3. COBIT (*Control Objectives for Information and related Technology*)
    - Marco de trabajo de buenas prácticas para TI
    - Desarrollado por la Information Systems Audit and Control Association
    - Última versión: 5 (junio 2012)
    - Incluye 34 objetivos de alto nivel, que cubren 215 objetivos de control categorizados en 4 dominios: planificación y organización, adquisición e implementación, entrega y soporte, y monitorización y evaluación.

Tema 2: Introducción a los sistemas Linux/Unix

Administración de Sistemas e Redes

Tomás Fernández Pena

tf.pena@usc.es

Índice

1. Introducción a Unix y Linux

Características de UNIX:

- Sistema operativo potente, flexible y versátil.
- Características: portabilidad, adaptabilidad y simplicidad, naturaleza **multiusuario** y **multitarea**, adecuación a redes.
- Disponibilidad de código fuente (algunas versiones)
- Implementado casi íntegramente en C (lenguaje de alto nivel).

GNU/Linux:

- Sistema operativo libre, de código abierto, similar a Unix
- Código fuente con licencia GPL
- Disponible para un gran número y variedad de sistemas: supercomputadores, servidores, sobremesas, portátiles, PDAs, móviles, sistemas empotrados,...

1.1. Historia de Unix

- Multics: proyecto de Bell Labs (AT&T), General Electrics y el MIT (1969) para el sistema GE 645
  - demasiado ambicioso para la época (pobre rendimiento)
- Thompson y Ritchie (Bell) migran un juego (*Space Travel*) en Multics de GE 645 a PDP-7.
- Empiezan del desarrollo de un SO para el PDP-7 $\longrightarrow$ Surge UNIX
- En 1970, UNIX se instala en una PDP-11
- En 1971 se edita el primer *UNIX Programmer's Manual.*
- En 1973 UNIX se reprograma en C (Ritchie)
- En 1974/75 UNIX v6 se difunde fuera de los laboratorios Bell y llega a las universidades
  - Los investigadores tienen acceso al código fuente del UNIX de AT&T
- En 1977 la Universidad de Berkeley licencia UNIX BSD
- AT&T limita la distribución del código de UNIX a partir de la v7
  - se dificulta el acceso al código fuente
  - System III: primera versión comercial de UNIX (1982)
- Dos líneas principales: System V y BSD

AT&T System V

- A partir de UNIX Versión 6 y 7, AT&T lanza, en 1982, la primer versión de la linea comercial de UNIX: System III
- SysIII carecía de innovaciones como `vi` y `csh`
- En 1983 surge System V. Incluía algunas características de los sistemas BSD (p.e. `vi`, `curses`,...)
- En 1984 surge la SysV Release 2 y en 1987 la SVR3
- Finalmente, SysV Release 4 aparece en 1988
- SVR4 combina SVR3, 4.3BSD, XENIX (Microsoft), SunOS (Sun Microsystems) y agrega nuevas utilidades

Berkeley System Distribution

- Thompson, Bill Joy (co-fundador de Sun) y Chuck Haley (1975).
- Second Berkeley Software Distribution (2BSD), 1978, incorpora el editor vi (versión visual de ex) y el C shell.
- En 1979, 3BSD, combina 2BSD con UNIX v7.
- DARPA (*Defense Advanced Research Projects Agency*) colabora con las nuevas versiones 4BSD: 4.1BSD, 4.2BSD y en 1986 4.3BSD (implementación de TCP/IP).
- Conflicto con AT&T por el uso de código propietario.
- Su última versión es 4.4BSD-Lite Rel. 2 (1995), sin código propietario AT&T. En ella se basan muchas variantes:
    - FreeBSD, OpenBSD, NetBSD, Darwin (base de OS X e iOS), etc.

Otras versiones

La mayoría de los UNIX actuales derivan de System V o BSD, o son una mezcla de los dos

- XENIX: desarrollada por Microsoft en 1980 para uso en microprocesadores, derivada del AT&T UNIX v7
- Openserver (antes SCO UNIX): derivada de XENIX y desarrollada por *Santa Cruz Operation*, hoy propiedad de Xinuos
- UnixWare: desarrollado por Novell a partir de System V, ahora propiedad de Xinuos
- SunOS: desarrollado por Sun Microsystems (ahora Oracle), en 1982, basado en BSD
- Mach: microkernel desarrollado en la Carnegie-Mellon University, basado en 4.3BSD
- XNU: kernel basado en Match, que forma parte de Darwin
- OSF/1 (Open Software Foundation): DEC, IBM y HP desarrollan un UNIX para competir con System V y SunOS:
    - Basado en el kernel Mach

    - Llamado después Digital UNIX y ahora Tru64

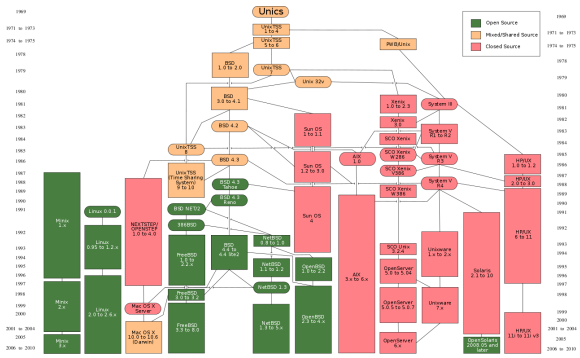
- GNU Hurd: conjunto de servicios que corren encima de GNU Mach formando el kernel del SO de GNU
- Minix: escrito por Andrew S. Tanenbaum de la Vrije Universiteit, para correr en los IBM PCs
- Linux: desarrollado por Linus Torvals, primera versión en 1991

Versiones comerciales

Muchas empresas tienen versiones propietarias:

- Oracle SUN: Oracle Solaris (evolución de SunOS versión 5 y SVR4), versiones para Sparc y x86, última versión Solaris 11 (versiones *open source* OpenSolaris, derivado OpenIndiana)
- IBM: AIX (*Advanced Interactive eXecutive*) para servidores IBM, basado en OSF/1 y SVR4, última versión AIX 7.1
- HP: HP-UX, versiones para PA-RISC e Itanium, variante System V con características de OSF/1, última versión 11i
- SGI: IRIX basado en System V con extensiones BSD, para sistemas MIPS; última versión 6.5 (2006)
- Apple: Mac OS X, con dos partes Darwin + Aqua (GUI); Darwin basado en Mach y BSD

Evolución de UNIX



Más detalles en http://www.levenez.com/unix/

1.2. Sistemas GNU/Linux

Linux:

1. En agosto de 1991, el estudiante finlandés Linus Torvals, presenta en Internet la versión 0.01 del kernel de un nuevo SO, inspirado en MINIX (aunque sin código de MINIX)
    - Esta primera versión tenía poco más de 10.000 líneas de código
2. En 1992, Linux se libera bajo licencia GPL
3. A través de Internet, muchos programadores se unieron al proyecto
4. En 1994 Linux alcanzó la versión 1.0
5. En 2003, llegamos a la versión 2.6, con casi 6 millones de líneas de código
6. En 2011, versión 3.0

GNU:

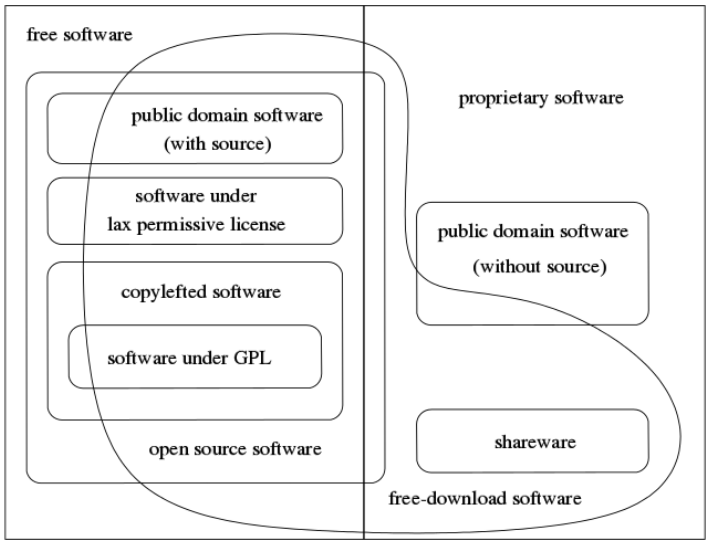
- El proyecto GNU (*GNU's Not Unix*) fue iniciado en 1983 por Richard Stallman bajo los auspicios de la Free Software Foundation
    - Objetivo: crear un sistema operativo completo basado en *software libre*, incluyendo herramientas de desarrollo de software y aplicaciones
- En el momento de la liberación, GNU no tenía listo su kernel
    - Linux fue adaptado para trabajar con las aplicaciones de GNU: Sistema GNU/Linux
        1. Kernel Linux +
        2. Aplicaciones GNU: compilador (gcc), librería C (glibc) y depurador (gdb), shell bash, GNU Emacs, GNOME, Gimp,...
    - GNU tiene ahora su propio kernel: GNU Hurd

Mascotas

Características de Linux

1. Sistema operativo de código abierto, multitarea y multiusuario
2. Portable (corre en arquitecturas Intel x86 y IA64, Sparc, MIPS, PowerPC, Alpha, PARisc,...)
3. Soporte para multiprocesador
4. Soporte para múltiples sistemas de ficheros
5. Kernel de tipo monolítico con módulos cargables dinámicamente

Software Libre y Open Source



Software libre (*free software*):

- Movimiento que parte de las ideas de Richard Stallman
- El software, una vez obtenido puede ser usado, copiado, estudiado, modificado y redistribuido
- La distribución no tiene que ser necesariamente gratuita

Open Source (o software de código abierto):

- Posibilidad de acceder al código fuente, y modificarlo y distribuirlo dentro de una determinada licencia de código abierto (ver www.opensource.org/licenses)

- La Open Source Initiative fue fundada en febrero de 1998 por Bruce Perens y Eric S. Raymond para la certificación de software Open Source

FLOSS Free/Libre/Open-Source Software

- Software libre y open software

Diferencia entre ellos principalmente filosóficas

- Código abierto: es una metodología de programación
- Software libre: asociado a la libertad del usuario

Ejemplo de la diferencia: dispositivos *tiranos* o *tivoized*
Más información: www.gnu.org/philosophy/

Licencia GPL

La licencia GPL (GNU General Public License) :

1. Bajo GPL el software puede ser copiado y modificado
2. Las modificaciones deben hacerse públicas bajo GPL (copyleft)
3. Se impide que el código se mezcle con código propietario

La licencia LGPL (GNU Lesser General Public License) permite integrar el software con software propietario

- Pensado para librerías que pueden ser usadas en el desarrollo de software propietario

Más información sobre licencias:

- Introducción a las licencias
- Varias licencias y comentarios

Distribuciones de GNU/Linux

Colección de software que forma un S.O. basado en el kernel Linux; normalmente incluye:

1. El kernel Linux
2. Las aplicaciones GNU (o parte de ellas)

3. Software de terceros, libre o propietario: X Windows, servidores, utilidades,...

Las distribuciones difieren en el empaquetado de los programas (RPM, deb, tgz), el programa de instalación y herramientas específicas

- Lista de distribuciones en wikipedia: en.wikipedia.org/wiki/List_of_Linux_distributions
- Timeline de distribuciones
- Información interesante en http://www.distrowatch.com

Algunas de las más populares son Debian, Red Hat (Fedora), Mandriva (Mageia), Slackware, SuSE, Gentoo, Ubuntu...

Debian

- Distribución totalmente libre, sin fines comerciales
- Tres ramas en la distribución:
  1. *Stable*: destinada a entornos de producción (desde mayo 2013, versión 7.0 *wheezy*)
  2. *Testing*: software más nuevo, en fase de prueba (actualmente *jessie*)
  3. *Unstable*: en fase de desarrollo (siempre *sid*)
- Versiones anteriores:
  - 6.0 squeeze, febrero 2011
  - 5.0 lenny, febrero 2009
  - 4.0 etch, abril 2007
  - 3.1 sarge, junio 2005
  - 3.0 woody, julio 2002
  - 2.2 potato, agosto 2000
  - 2.1 slink, marzo 1999
  - 2.0 hamm, julio 1998
  - 1.3 bo, junio 1997
  - 1.2 rex, diciembre 1996
  - 1.1 buzz, junio 1996
- Algunas características
  1. Gran número de aplicaciones disponibles: más de 35000 paquetes
  2. Potente formato de empaquetado: paquetes DEB y herramienta APT
  3. Instalación y cambio de versiones a través de red

Ubuntu

- Distribución enfocada a ordenadores de escritorio (*Desktop Computers*), aunque existe la versión para servidores
- Basada en Debian, Ubuntu concentra su objetivo en la usabilidad, lanzamientos regulares y facilidad en la instalación
- Patrocinado por Canonical Ltd., una empresa privada fundada y financiada por el empresario sudafricano Mark Shuttleworth
- Última versión: Ubuntu 13.04 (*Raring Ringtail*), fue lanzada el 25 de abril de 2013
- Próxima versión: Ubuntu 13.10 (*Saucy Salamander*) prevista para el 17 de octubre de 2013
- Proyectos relacionados: kubuntu, edubuntu, xubuntu

Red Hat

- Una de las principales firmas comerciales del mundo GNU/Linux
- Fundada por Marc Ewing y Bob Young en 1994
- Inicialmente, proporcionaba distribuciones para el usuario individual (versiones personal y profesional), y orientadas a empresas (versión Enterprise)
- Introduce el formato de empaquetado RPM (*RedHat Package Manager*)
- Desde 2002, orientado en exclusiva al mercado corporativo
  - Cede la última distribución personal (RH 9) a la comunidad $\longrightarrow$ aparece el proyecto Fedora
- Última versión: Red Hat Enterprise Linux 6 (*Santiago*) desde noviembre de 2010
- Próxima versión: Red Hat Enterprise Linux 7, prevista final 2013

- Distribuciones libres que clonan RHEL: CentOS, Scientific Linux, ClearOS, etc.

Fedora

- Objetivo: construir un SO completo, de propósito general basado exclusivamente en código abierto
- Parte de la versión Red Hat 9
- Mantiene el sistema de paquetes RPM
- Última versión: Fedora 19 (*Schrödinger's Cat*), 2 de julio de 2013

Slackware

- Una de las primeras distribuciones: creada en 1993 Patrick Volkerding
- Orientada hacia usuarios avanzados:
- Ultima versión estable: Slackware 14.0 (28 de septiembre de 2012)

SuSE Linux

- Compañía alemana fundada en 1992, subsidiaria de *The Attachmate Group*
- Originalmente basada en Slackware
- Herramienta de configuración gráfica: YaST (*Yet Another Setup Tool*)
- Principales versiones: SUSE Linux Enterprise Server y SUSE Linux Enterprise Desktop
- Versión 100 % open source: openSUSE, última versión 12.3 (marzo 2013)

Gentoo Linux

- Distribución orientada a permitir la máxima adaptabilidad y rendimiento
    - puede ser optimizada y configurada automáticamente para el uso en un sistema concreto
- Portage: Sistema de distribución, compilación e instalación de software

Otras distribuciones

- Existen cientos de distribuciones diferentes de Linux
  - Adaptadas a diferentes necesidades: seguridad, multimedia, sistemas viejos, análisis forense, clusters...
  - Muchas tienen versiones Live
  - Suelen estar basadas en las principales distribuciones
- Ejemplos (ver distrowatch.com):
  1. Sistemas basados en Debian/Ubuntu: Ubuntu-USC, LinuxMint, Knoppix y derivados (STD, Damn Small...), trisquel, Guadalinex, ...
  2. Sistemas basados en RedHat/Fedora: Mageia, PCLinuxOS, Oracle Linux, Springdale, Berry Linux, kororaa, Tinyme, Rocks...
  3. Sistemas basados en Gentoo: Funtoo, Sabayon, Pentoo, Toorox...
  4. Sistemas basados en Slackware: SLAX, Zenwalk, Vectorlinux, Porteus, Absolute...

2. Instalación del sistema y de software

A la hora de instalar un sistema, tenemos que tener en cuenta el tipo de funciones que va a desempeñar.
Podemos distinguir:

1. Sistema de escritorio: usado en tareas rutinarias (ofimática, acceso a Internet, etc.)
2. Estación de trabajo (*workstation*): sistema de alto rendimiento, generalmente orientado a una tarea específica
   - estación dedicada al cálculo (p.e. aplicaciones científicas)
   - estaciones gráficas (p.e. diseño 3D)
3. Servidores: ofrecen servicios a otras máquinas de la red
   - servicios de disco, impresión, acceso a Internet, filtrado, etc.

2.1. Tipos de servicios

Un sistema servidor ofrece servicios al resto de sistemas de la red:

1. Aplicaciones
   - servicios de terminales, conexión remota (telnet, ssh), aplicaciones gráficas a través de X Window, aplicaciones web, etc.
2. Ficheros
   - acceso a ficheros a través de FTP,
   - servicio transparente a través de NFS o Samba
3. Impresión
   - servir impresoras locales o remotas a otros sistemas UNIX o Windows
4. Servicios de información de red, por ejemplo, NIS, NIS+ o LDAP
   - permiten centralizar la información de las máquinas, usuarios y recursos
5. Servicios de configuración dinámica de máquinas
   - DHCP (*Dynamic Host Configuration Protocol*): permite configurar dinámicamente la red de los clientes
6. Correo electrónico
   - agentes MTA (*Mail Transfer Agent*) para recuperar y retransmitir correo, o servicios de POP o IMAP
7. Servidor Web (p.e. Apache)
8. Servicio de nombres (DNS)
9. Servicio de base de datos
10. Servicios de acceso a Internet: NAT, proxy
11. Servicios de filtrado (firewall)

2.2. Virtualización

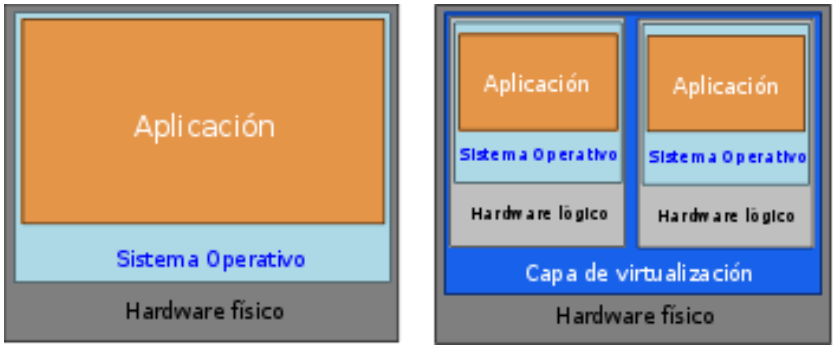
Abstracción de un conjunto de recursos computacionales para que puedan ser utilizados de forma más conveniente

- Memoria virtual
- Sistemas RAID o LVM
- Virtualización de servidores

Virtualización de servidores

- Máquina virtual
  - Entorno virtual entre el sistema real y el usuario final que permite que este ejecute un software determinado
  - Normalmente usado para ejecutar varios sistemas operativos simultaneamente sobre el mismo hardware
- Usos de la virtualización
  - Consolidación de servidores
  - Execución de aplicacións non-fiables
  - Recuperación de desastres
  - Pruebas y desarrollo de software
  - Computación elástica (cloud computing)

Conceptos:



- Sistema anfitrión (*host*): SO ejecutado sobre la máquina real
- Sistema huesped (*guest*): SO ejecutado sobre la máquina virtual

Algunas herramientas de virtualización:

- VirtualBox desarrollado originalmente por la empresa alemana Innotek, ahora propiedad de Oracle; version Open Source (VBox OSE) y propietaria
- QEMU emulador/virtualizador de código abierto desarrollado por Fabrice Bellard
- KVM virtualización asistida por hardware, utiliza una versión modificada de QEMU como front-end.
- Xen desarrollado inicialmente en la universidad de Cambridge, versiones comerciales Citrix XenServer, Oracle VM,...
- VMWare Workstation programa propietario de VMware Inc.; es uno de los más conocidos (versiones para Windows y Linux)
- Virtual PC herramienta de Microsoft; versiones para Windows y Mac

Una comparativa en wikipedia
Tipos de virtualización:

- Emulación (o recompilación dinámica): la máquina virtual simula el hardware completo
  - Permite ejecutar SOs para sistemas diferentes del anfitrión
  - Normalmente es lenta
  - Ejemplos: Bochs, PearPC, QEMU sin aceleración,...
- Paravirtualización: la máquina virtual no simula todo el hardware, sino que ofrece una API especial
  - Requiere modificacións en el SO huesped
  - Velocidad nativa
  - Ejemplos: Xen
- Virtualización completa: la máquina virtual sólo simula el hardware necesario para permitir que un SO huesped se pueda ejecutar
  - El SO huesped debe ser para el tipo de arquitectura del host
  - Velocidade cerca de la nativa

- Exemplos: VMWare, QEMU con aceleración, Parallels Desktop for Mac, VirtualPC para Windows, etc.

- Virtualización asistida por hardware
  - El hardware del anfitrión proporciona soporte para mejorar la virtualización: x86 virtualization, en Intel VT ou AMD-V
  - Velocidad similar a la paravirtualización sin necesidad de modificar el huesped
  - Ejemplos: Xen, VirtualBox, KVM, VMWare, Parallels Workstation, etc.
- Virtualización a nivel de SO: aisla varios servidores sobre el SO anfitrión
  - Los SO huespedes son los mesmos que el anfitrión, ya que usan el mesmo kernel
  - Ejemplos: User-mode Linux, FreeBSD Jail, Linux-VServer, Virtuozzo,...

2.3. Instalación de Linux Debian

Para detalles de instalación ver Guía de instalación de Debian

- Descargaremos la imagen de CD pequeño (fichero `debian-7.1.0-i386-netinst.iso`)

- `Enter` para iniciar con opciones por defecto, `Advances options` para opciones de instalación avanzadas, `Help` para ayuda

Siguientes pasos en la instalación[1]

- Selección de idioma y teclado
- Configuración de la red
  - Por defecto, intenta configurarla por DHCP
  - Si no lo consigue, pasa a configuración manual (indicar IP, máscara, pasarela y DNSs)
- Poner un nombre a la máquina e indicar el dominio (si alguno)
- Fijar el password del superusuario (root) y crear un usuario no privilegiado

Cuenta del superusuario

- El superusuario es un usuario especial que actúa como administrador del sistema
  - Tiene acceso a todos los archivos y directorios del sistema
  - Tiene capacidad para crear nuevos usuarios o eliminar usuarios
  - Tiene capacidad de instalar y borrar software del sistema o aplicaciones
  - Puede detener cualquier proceso que se está ejecutando en el sistema
  - Tiene capacidad de detener y reiniciar el sistema
- El login del superusuario es `root` (aunque puede cambiarse)
- No es conveniente acceder al sistema directamente como root:
  - acceder como un usuario sin privilegios, y
  - obtener los permisos de root haciendo `su` (necesitamos la contraseña de root)

[1] En cualquier momento de la instalación tenemos acceso a una consola pulsando `Alt-F2`; usar `Alt-F1` para volver a la instalación

Elección de contraseña

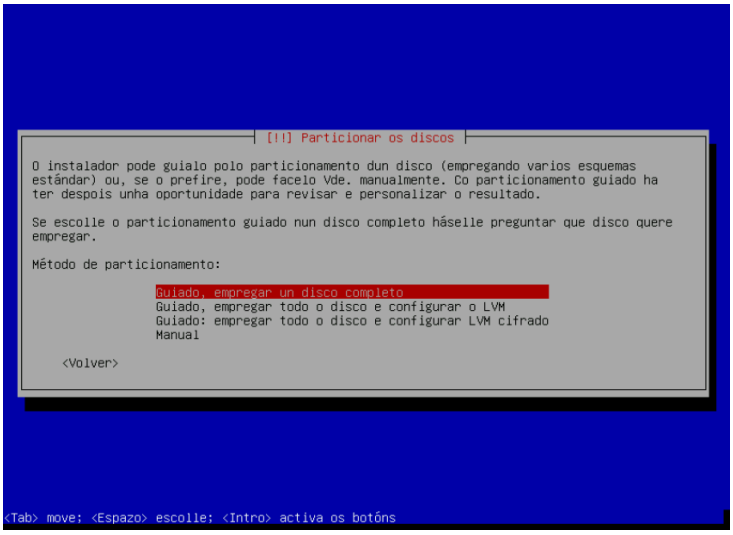
- Tener una contraseña de root adecuada es básico para la seguridad de un sistema
- Las contraseñas de usuario también deberían ser adecuadas
- Recomendaciones para elegir una contraseña:
  - No usar el nombre de usuario (login) ni variantes de este (p.e. login: pepe, passwd: pepe98)
  - No usar el nombre real del usuario ni los apellidos
  - No usar palabras contenidas en diccionarios, o palabras de uso común
  - Usar más de 6 caracteres para la contraseña
  - Mezclar caracteres en mayúsculas y minúsculas, con caracteres no alfabéticos (números, signos de puntuación, etc.)
  - Usar contraseñas fáciles de recordar, para evitar tener que apuntarlas
  - Cambiar la contraseña con frecuencia (p.e. una vez al mes)
- La contraseña se cambia con el comando `passwd`
  - `passwd`: cambia la contraseña (password) del usuario
  - Ejemplo: usuario pepe

```
# passwd
Changing password for pepe
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Continuación de la instalación

En una instalación por red los paquetes se traen de un repositorio remoto a través de http o ftp

- Seleccionar el huso horario
- Realizar el particionado del disco (modo guiado o manual)



Particionado del disco

Podemos optar por instalar todo el sistema en una sola partición, aunque no es nada recomendable

- preferible instalar diferentes directorios del sistema en diferentes particiones
- la estructura de directorios UNIX sigue el estándar FHS (*Filesystem Hierarchy Standard*)

Filesystem Hierarchy Standard

Localización estándar de los ficheros

- `/bin/` (***bin****aries*) - ejecutables esenciales (`ls`, `cat`, `bash`, etc.)
- `/sbin/` - (***su****peruser* ***bin****aries*) - ejecutables esenciales para el superusuario (`init`, `ifconfig`, etc.)
- `/lib/` - Librerías esenciales para los ejecutables en `/bin/` y `/sbin/`
- `/usr/` (***U****nix* ***s****ystem* ***r****esources*) - aplicaciones y código fuente usados por los usuarios y el superusuario
  - `/usr/bin/` - más aplicaciones de usuario
  - `/usr/sbin/` - más aplicaciones para el superusuario
  - `/usr/lib/` - librerías esenciales para los ejecutables en `/usr/bin/` y `/usr/sbin/`

    - `/usr/share/` - datos, independientes de la arquitectura, necesarios para las aplicaciones y páginas de manual (`/usr/share/man`, `/usr/share/info`)
    - `/usr/include/` - ficheros de cabecera (`.h`) estándar
    - `/usr/src/` (opcional) - código fuente (del kernel u otras aplicaciones)
    - `/usr/X11R6/` (opcional) - sistema X Window, versión 11 release 6
    - `/usr/local/` - aplicaciones que no son parte del sistema operativo
- `/etc/` - contiene muchos de los scripts y ficheros de configuración del sistema
    - `/etc/X11/` (opcional) - configuración de X Window
    - `/etc/skel/` (opcional) - ficheros de configuración para los usuarios
- `/var/` - ficheros **var**iables (logs, bases de datos, etc.)
    - `/var/log/` - ficheros de log
    - `/var/spool/` - ficheros temporales de impresión, e-mail y otros
- `/tmp/` - ficheros temporales
- `/opt/` - otras aplicaciones software (estáticas)
- `/srv/` - datos de servicios proporcionados por el sistema (páginas web, ftp, cvs, etc.)
- `/boot/` - ficheros usados por el gestor de arranque, incluyendo el kernel

Otros directorios del sistema

- `/` - directorio raíz del sistema
- `/home/` (opcional) - directorio de usuarios (directorio inicial o *home*)
- `/root/` (opcional) - directorio *home* del superusuario
- `/dev/` - ficheros de acceso a periféricos
- `/proc/` - directorio virtual conteniendo información del sistema
- `/sys/` - similar a `/proc`, contiene información de dispositivos (sólo kernel 2.6)

- `/media/` - punto de montaje para medios removibles
- `/mnt/` - punto de montaje para sistemas temporales

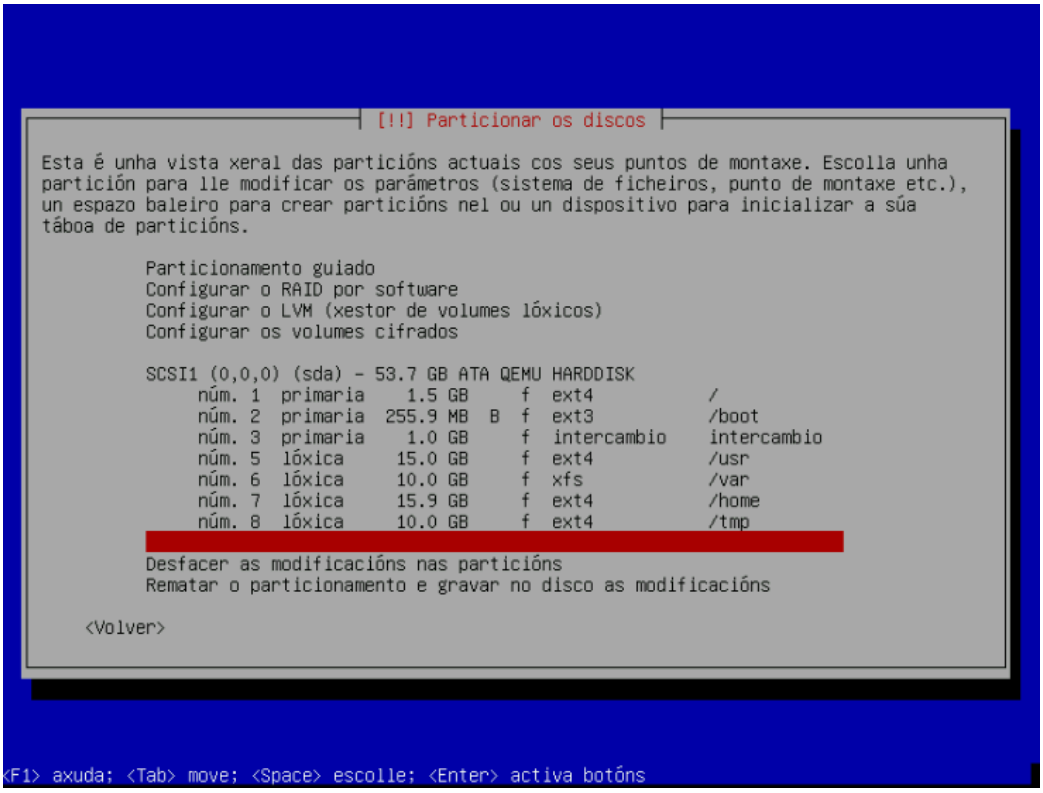
Para más información ver www.pathname.com/fhs/

Esquemas de particionamiento

Dependiendo del tipo de sistema podemos escoger diferentes esquemas de particionamiento, algunos ejemplos:

- Maquina de escritorio (un sólo usuario), tres particiones
  - `swap` - área de intercambio; siempre necesaria, tamaño función del tamaño de la RAM y del tipo de aplicaciones que se ejecuten (como orientación, tomar al menos el doble de la RAM)
  - `/home/` - disco de los usuarios, tamaño en función de las necesidades del usuario
  - `/` - resto del disco
- Sistema multiusuario, además de las particiones anteriores crear particiones separadas para `/usr`, `/var` y `/tmp`
  - `/usr` podría montarse en modo sólo-lectura después de que todo el sistema esté instalado (dificulta la introducción de Troyanos)
  - tener `/var` y `/tmp` en su partición evita que un usuario llene todo el disco
- Particiones adicionales:
  - `/boot` - en versiones antiguas de Linux se necesitaba que el directorio `/boot/` estuviese por debajo del cilindro 1024
  - `/chroot` - para aplicaciones en un entorno *enjaulado* (p.e. DNS, Apache, etc.)
  - `/var/lib` - partición para gestionar ficheros del servidor de bases de datos o del proxy (MySQL, squid) (limitar la posibilidad de un ataque por denegación de servicio)

Ejemplo de partición (disco de 50 G):


[!!] Particionar os discos
Esta é unha vista xeral das particións actuais cos seus puntos de montaxe. Escolla unha
partición para lle modificar os parámetros (sistema de ficheiros, punto de montaxe etc.),
un espazo baleiro para crear particións nel ou un dispositivo para inicializar a súa
táboa de particións.
Particionamento guiado
Configurar o RAID por software
Configurar o LVM (xestor de volumes lóxicos)
Configurar os volumes cifrados
SCSI1 (0,0,0) (sda) - 53.7 GB ATA QEMU HARDDISK
núm. 1 primaria 1.5 GB f ext4 /
núm. 2 primaria 255.9 MB B f ext3 /boot
núm. 3 primaria 1.0 GB f intercambio intercambio
núm. 5 lóxica 15.0 GB f ext4 /usr
núm. 6 lóxica 10.0 GB f xfs /var
núm. 7 lóxica 15.9 GB f ext4 /home
núm. 8 lóxica 10.0 GB f ext4 /tmp
Desfacer as modificacións nas particións
Rematar o particionamento e gravar no disco as modificacións
<Volver>
<F1> axuda; <Tab> move; <Space> escolle; <Enter> activa botóns

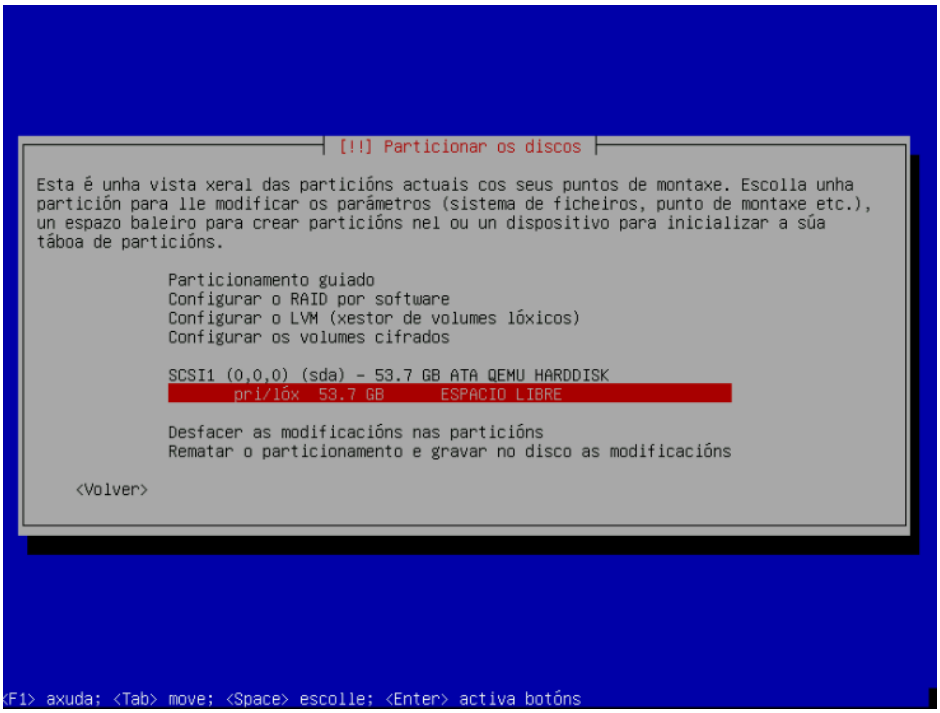
Particionamiento durante la instalación

Dos opciones:

- Particionamiento guiado (con o sin LVM)
  - Selecciona el tamaño de las particiones de manera automática
- Particionamiento manual
  - Particionamiento manual
    - control total del número y tamaño de las particiones

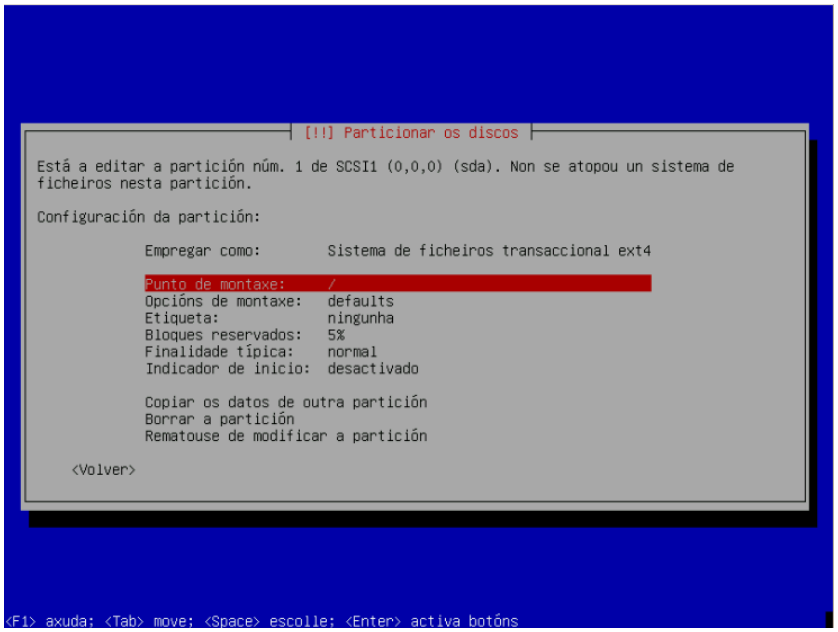
Particionamiento manual

1. Seleccionamos el disco a particionar y crear nueva tabla de particiones:



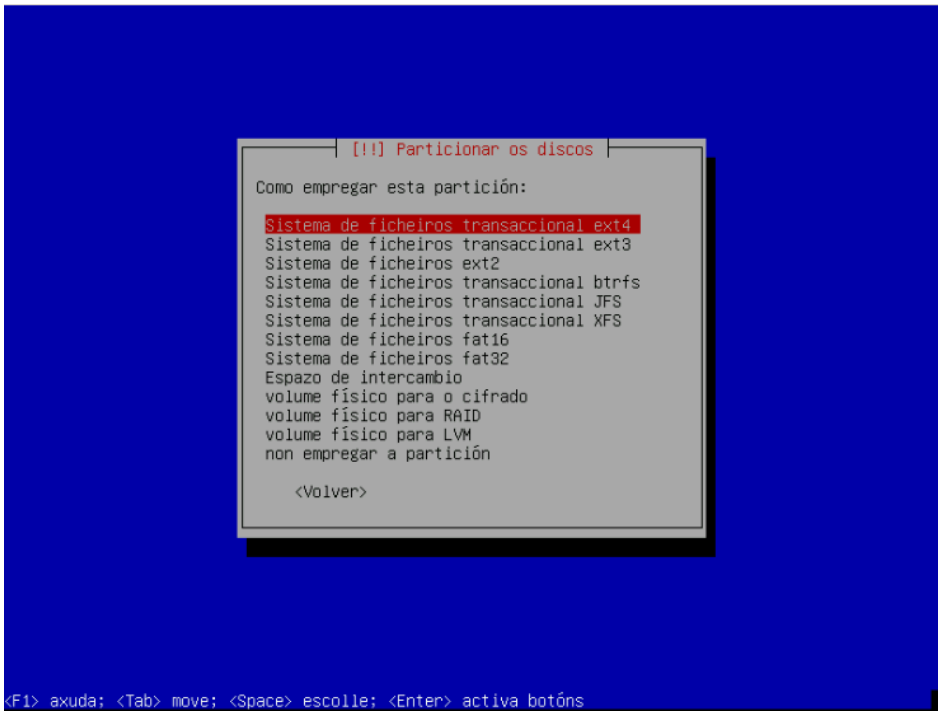
2. Creamos una nueva partición indicándole el tamaño, el tipo (primaria o lógica) y la localización (comienzo o final)
   - puede haber 4 primarias o 3 primarias y una extendida, que se puede dividir en varias lógicas

Ejemplo de partición para /



Sistemas de ficheros

Linux soporta múltiples sistemas de ficheros
Para cada partición podemos seleccionar los siguientes:



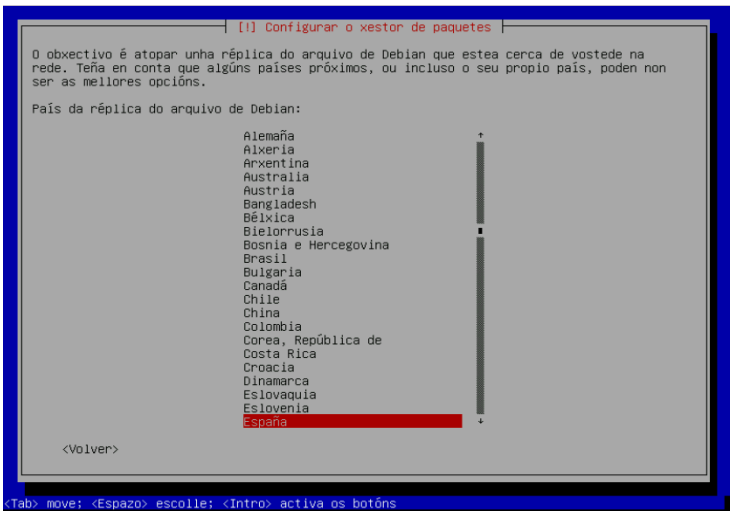
- `ext2` - *Second EXTended filesystem*, sistema estándar Linux
- `ext3` - *Third EXTended filesystem*, versión con *journal* de ext2, que evita corrupción (opción por defecto)
  - es posible convertir ext2 en ext3 con el comando *tune2fs -j*
  - muy robusto, aunque no escala muy bien (no ideal para filesystems muy grandes, ficheros muy grandes o un número de ficheros en un directorio muy alto)
- `ext4` - *Fourth EXTended filesystem*, última versión, disponible en el kernel 2.6.28, mejoras en velocidad y otros aspectos
- `ReiserFS`, `JFS`, `XFS` - otros tipos de sistemas transaccionales (con *journal*) usados en diferentes sistemas
  - `ReiserFS` - por defecto en algunas distribuciones Linux (p.e. Slackware)
    - mayor rendimiento que `ext2` y `ext3`, principalmente con ficheros pequeños (menos de 4k) y buena escalabilidad
    - Sucesor: Reiser4
  - `XFS` - usado en sistemas SGI Irix

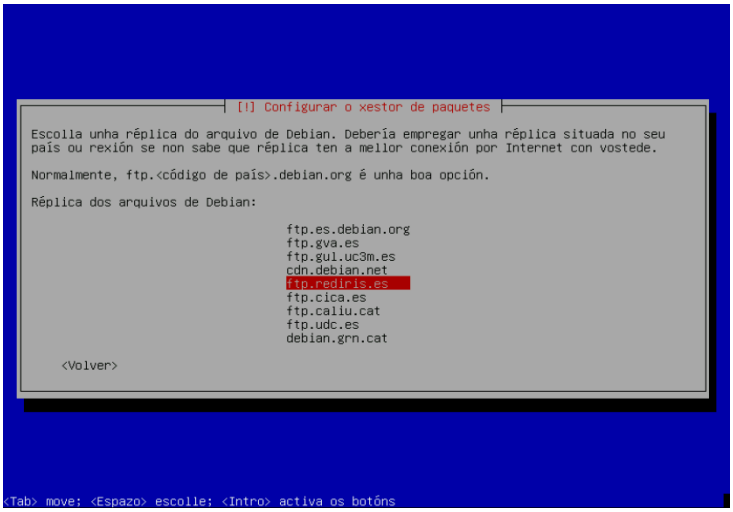
    - optimizado para escalabilidad
    - recomendado en grandes sistemas SCSI o *fiber channel* con fuente de alimentación ininterrumpida (utiliza caché de forma agresiva $\longrightarrow$ perdida de datos si el sistema se apaga)
  - `JFS` - usado en máquinas de IBM
- `fat16`, `fat32` - usados en MS-DOS y Windows 95/98/Me

Comparativa en wikipedia

Últimos pasos en la instalación

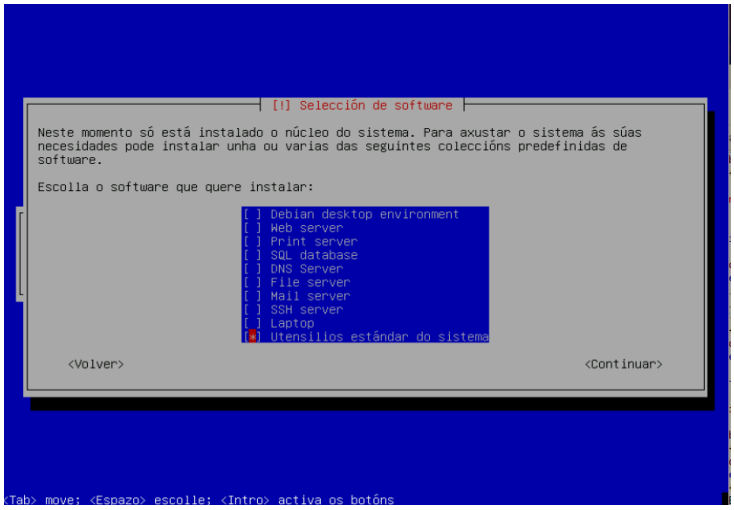
- Debemos seleccionar el *mirror* desde el que descargar el software
  - Existen varios repositorios de paquetes Debian $\longrightarrow$ elegir el más cercano
  - Introducir la información del proxy, en caso de ser necesario





- Seleccionar los paquetes software a instalar
- Instalar del gestor de arranque

Selección de paquetes



- Elegir los paquetes a instalar:
  - aunque optemos por no instalar nada, se instalarán todos los paquetes con prioridad “estándar”, “importante” o “requerido” que aún no estén instalados en el sistema
- Podemos repetir este paso con el sistema instalado usando el comando `tasksel`

Instalación del gestor de arranque

Gestor de arranque: permite seleccionar el SO a arrancar
Existían 2 posibilidades en Linux

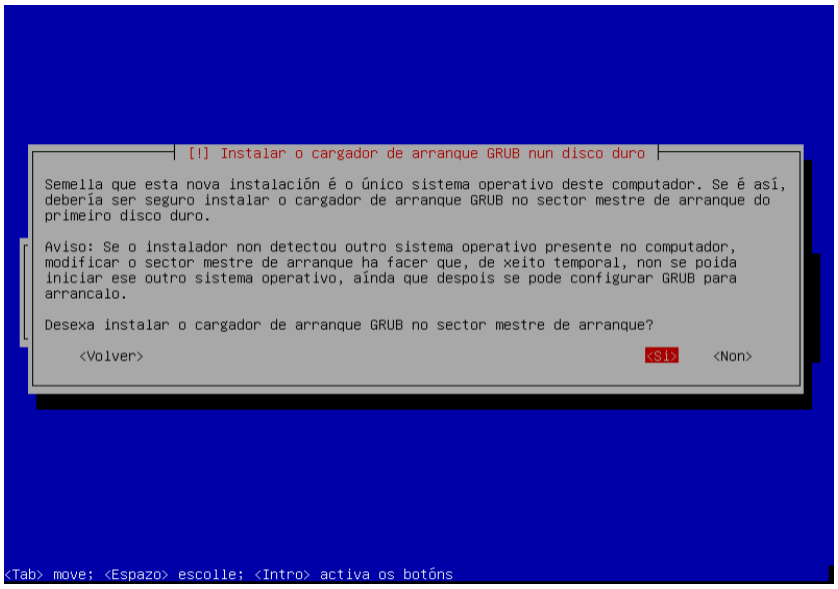
- LILO (*LInux Loader*), cargador clásico en Linux
- GRUB (*GRand Unified Bootloader*), cargador del proyecto GNU

La gran mayoría de las distribuciones usan GRUB (las más actuales la versión 2)
El gestor de arranque se carga normalmente en el MBR del primer disco

- MBR (*Master Boot Record*) está localizado en el primer sector del disco
- en el MBR se encuentra información sobre las particiones (*Master Partition Table*) y un pequeño código (*Master Boot Code*)
- cuando el sistema se inicia, la BIOS carga el *Master Boot Code*, que permite seleccionar el sistema a cargar, y transfiere el control al programa de arranque del SO (localizado en `/boot`)

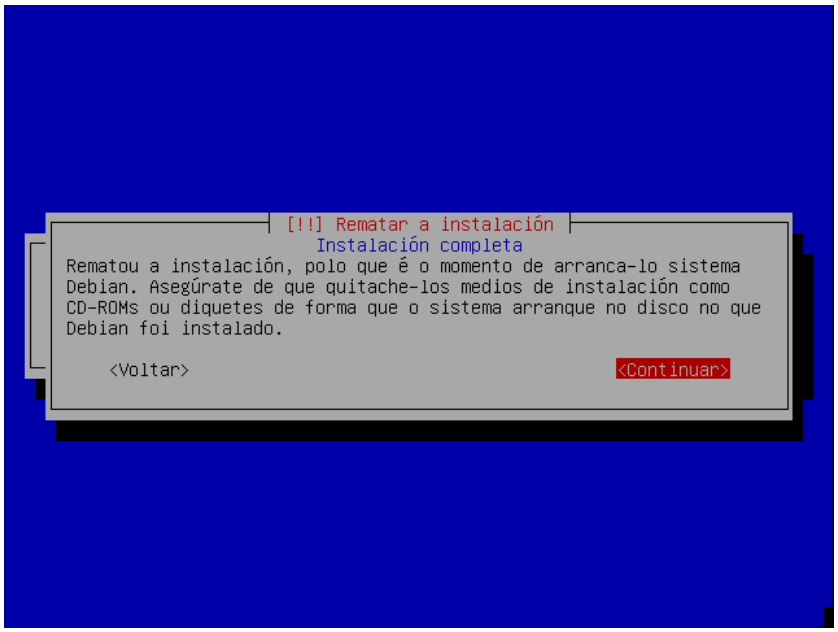
El gestor de arranque puede también cargarse en el primer sector de la partición root (por si tenemos otro bootloader en el MBR)

Instalación de GRUB en Debian



Finalización de la instalación

Debian: la instalación termina aquí



Debemos reiniciar el sistema para continuar

Logical Volume Management (LVM)

Proporciona una visión de alto nivel de los discos

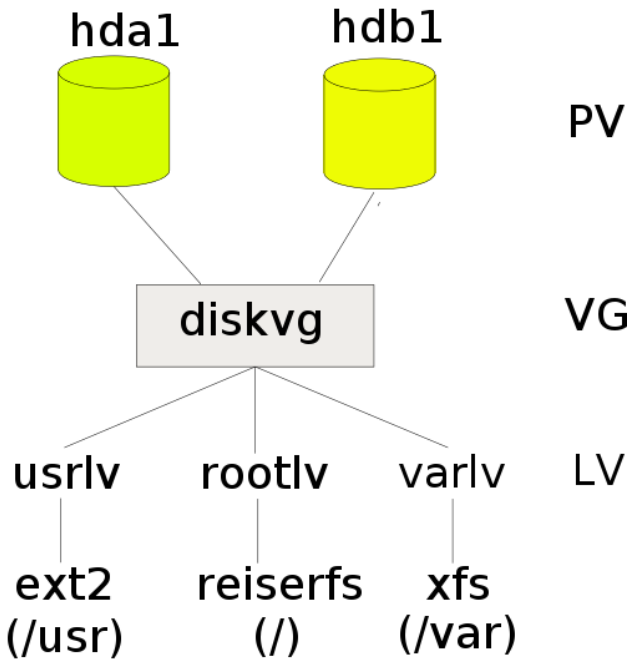
- permite ver varios discos como un único volumen lógico
- permite hacer cambios en las particiones sin necesidad de reiniciar el sistema
- permite gestionar los volúmenes en grupos definidos por el administrador

Conceptos (para más información LVM HOWTO):

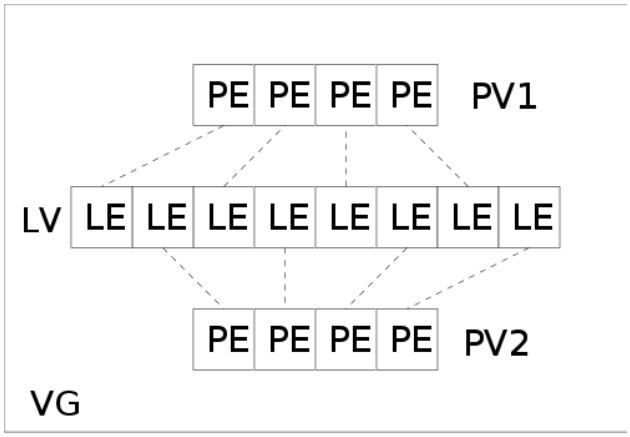
- **Volumen físico (PV)**: discos duros, particiones de los discos u otro dispositivo similar (p.e. RAID)
- **Volumen lógico (LV)**: *particiones* lógicas sobre las que se montan los sistemas de ficheros
- **Grupo de volúmenes (VG)**: agrupación de LV, que forman una unidad administrativa

- **Extensión física (PE)**: unidades básicas en las que se divide cada PV; el tamaño de cada PE es el mismo para todas los PV pertenecientes al mismo VG
- **Extensión lógica (LE)**: unidades básicas en las que se divide cada LV; para un VG el tamaño de las LE es el mismo que el de las PEs
- **Área de descripción del VG (DAVG)**: área donde se almacena la información (meta-data) sobre los LV y VG; sería el equivalente a la tabla de particiones de un sistema no-LVM

Estructura de LVM



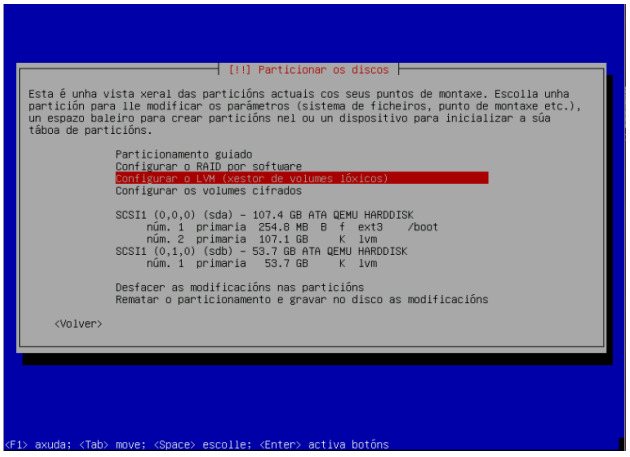
Hay una relación 1:1 entre cada LE y PE en un VG



- Se pueden elegir dos estrategias para mapear extensiones lógicas en extensiones físicas:
  - Mapeado Lineal: asigna un rango de PEs a un área de un LV en orden, por ejemplo LE 1-99 se mapean a PV1, y los LE 100-199 se mapean a PV2
  - Stripping: se reparten los LEs entre los distintos PVs
    - 1 LE → PV1[1], 2 LE → PV2[1], 3 LE → PV3[1], ...

Pasos para crear un sistema LVM
Suponemos un sistema con dos discos (`sda` y `sdb`)

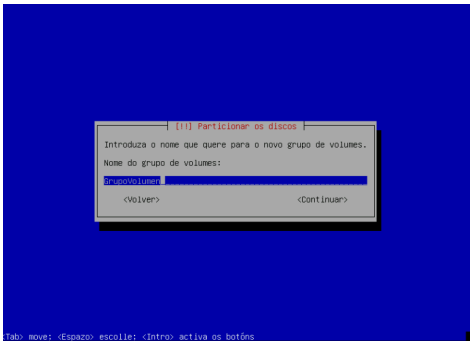
1. Crear los PV

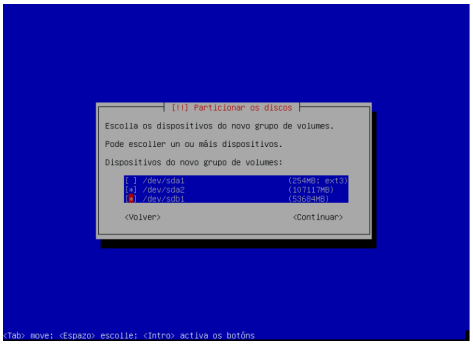


   - particionamos `sda` para reservar un espacio para `/boot` (dejamos `/boot` fuera de LVM para evitar problemas con el arranque, aunque en las últimas versiones no es necesario)

- definimos 2 volúmenes físicos
  - el primero incluye todo `sda` menos `/boot` (`sda2`)
  - el segundo incluye todo `sdb` (`sdb1`)

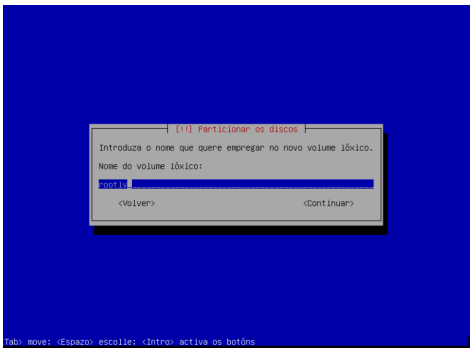
2. Crear un grupo de volumen que incluya los PVs

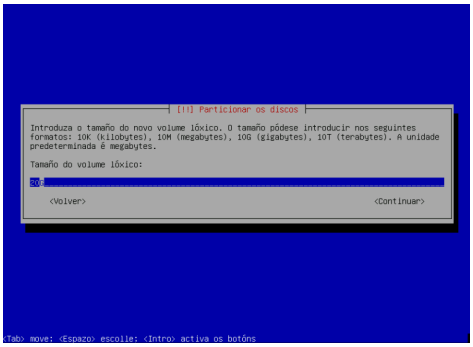




- podemos ponerle un nombre al grupo de volumen
- hacemos que incluya los dos volúmenes físicos que hemos definido en el punto anterior

3. Crear los volúmenes lógicos

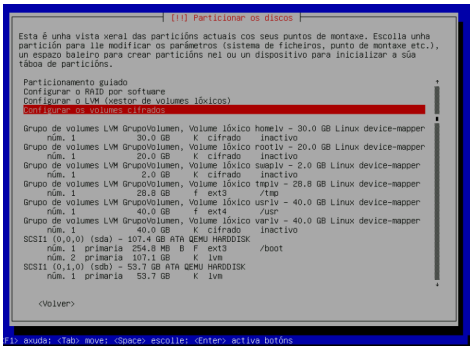


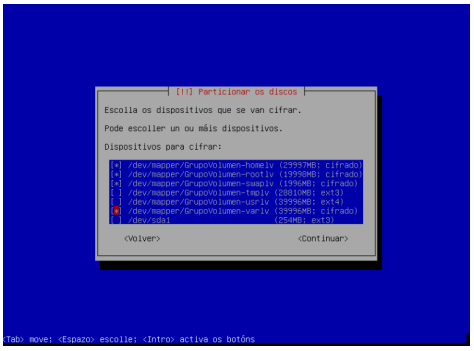


- creamos un volumen lógico por cada partición
- los LV pueden llevar un nombre identificativo

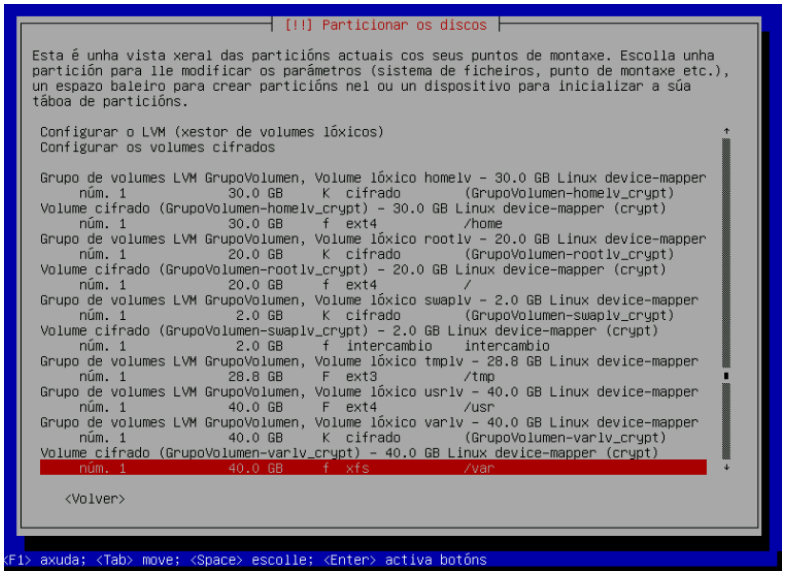
4. Cifrar sistemas de ficheros

- podemos usar algún LV como "volumen físico para cifrado"
- permite cifrar la información: contraseña para acceder a la misma





5. Asignar sistemas de ficheros a los volúmenes (cifrados o no)



Configuración del gestor de arranque

Debemos configurar GRUB para evitar que sea modificado el menu de arranque

- debemos usar una contraseña para el fichero de configuración `/boot/grub/menu.lst` para limitar:
  - la modificación de los parámetros iniciales
  - el acceso a determinadas imágenes
  - el acceso a modo monousuario

Más información sobre LILO y GRUB en

www.ac.usc.es/docencia/ASRI/Tema_4html/node19.html

2.4. Verificación de la instalación

- Las últimas distribuciones de Linux soportan la mayoría del hardware actual, incluyendo USB, Serial ATA, etc.
- Hay soporte Linux para múltiples arquitecturas: Intel, Alpha, MIPS, PowerPC, SPARC, etc.
- En el proceso de instalación se configura automáticamente casi todo el hardware
  - pueden tener algún problema con módems (*winmodens*) o impresoras no PostScript
  - con algunas tarjetas de vídeo puede ser difícil obtener aceleración 3D
- Más información en Linux Hardware Compatibility HOWTO o páginas relacionadas

Verificación del hardware

Para verificar los dispositivos PCI de nuestro sistema se puede usar `lspci`

- `lspci`: lista dispositivos PCI; algunas opciones (para más opciones `man lspci`):
  - `-v`: salida descriptiva
  - `-vv`: salida más descriptiva
  - `-t`: salida con estructura de árbol
- Ejemplo: sistema con discos IDE, tarjeta VGA y dos tarjetas de red:

```
sarge1:~# lspci
0000:00:00.0 Host bridge: Intel Corp. 440FX - 82441FX PMC [Natoma] (rev 02)
0000:00:01.0 ISA bridge: Intel Corp. 82371SB PIIX3 ISA [Natoma/Triton II]
0000:00:01.1 IDE interface: Intel Corp. 82371SB PIIX3 IDE [Natoma/Triton II]
0000:00:02.0 VGA compatible controller: Cirrus Logic GD 5446
0000:00:03.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL-8029(AS)
0000:00:04.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL-8029(AS)
```

- Ejemplo: sistema con PCI Express, discos SATA y varios hubs USB conectados

```
jumilla:~# lspci
0000:00:00.0 Host bridge: Intel Corp. 915G/P/GV Processor to I/O Controller (rev 04)
0000:00:01.0 PCI bridge: Intel Corp. 915G/P/GV PCI Express Root Port (rev 04)
0000:00:02.0 VGA compatible controller: Intel Corp. 82915G Express Chipset Family Graphics
Controller (rev 04)
0000:00:02.1 Display controller: Intel Corp. 82915G Express Chipset Family Graphics
Controller (rev 04)
0000:00:1c.0 PCI bridge: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) PCI Express Port 1
(rev 03)
0000:00:1c.1 PCI bridge: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) PCI Express Port 2
(rev 03)
0000:00:1d.0 USB Controller: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB UHCI #1
(rev 03)
0000:00:1d.1 USB Controller: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB UHCI #2
(rev 03)
0000:00:1d.2 USB Controller: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB UHCI #3
(rev 03)
0000:00:1d.3 USB Controller: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB UHCI #4
(rev 03)
0000:00:1d.7 USB Controller: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB2 EHCI
Controller (rev 03)
0000:00:1e.0 PCI bridge: Intel Corp. 82801 PCI Bridge (rev d3)
0000:00:1e.2 Multimedia audio controller: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family)
AC'97 Audio Controller (rev 03)
0000:00:1f.0 ISA bridge: Intel Corp. 82801FB/FR (ICH6/ICH6R) LPC Interface Bridge (rev 03)
0000:00:1f.1 IDE interface: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) IDE Controller
(rev 03)
0000:00:1f.2 IDE interface: Intel Corp. 82801FB/FW (ICH6/ICH6W) SATA Controller (rev 03)
0000:00:1f.3 SMBus: Intel Corp. 82801FB/FBM/FR/FW/FRW (ICH6 Family) SMBus Controller (rev
03)
0000:02:00.0 Ethernet controller: Broadcom Corporation NetXtreme BCM5751 Gigabit Ethernet
PCI Express (rev 01)
```

- Algunas definiciones:
  - UHCI: *Universal Host Controller Interface*, estándar de Intel para controladores USB (define como el controlador USB habla al ordenador y a su sistema operativo); otro estándar similar es OHCI (*Open Host Controller Interface*), desarrollado por Compaq, Microsoft y National Semiconductor Corp.
  - EHCI: *Enhaced Host Controller Interface*, versión extendida para USB 2
  - ICH6: *Intel I/O Controller Hub 6*: controlador para interfaz con el bus PCI
  - SATA: Serial ATA
  - SMBus: *System Management Bus*, bus sencillo para conectar dispositivos de bajo ancho de banda, usado para gestión de energía (p.e. control de batería en portátiles, sensores de temperatura, etc.)

Otro comando: `lsusb`

- `lsusb`: lista dispositivos USB; algunas opciones (para más opciones `man lsusb`):
  - `-v`: salida descriptiva
  - `-t`: salida con estructura de árbol
- Ejemplo: sistema con teclado, ratón, hubs USB y dos pendrive:

```
jumilla:~# lsusb
Bus 005 Device 019: ID 0c76:0005 JMTek, LLC. USBdisk
Bus 005 Device 015: ID 0424:a700 Standard Microsystems Corp.
Bus 005 Device 001: ID 0000:0000
Bus 004 Device 001: ID 0000:0000
Bus 003 Device 001: ID 0000:0000
Bus 002 Device 009: ID 413c:3010 Dell Computer Corp.
Bus 002 Device 001: ID 0000:0000
Bus 001 Device 011: ID 0ea0:2168 Ours Technology, Inc. Transcend JetFlash 2.0 / Astone USB
Drive
Bus 001 Device 007: ID 413c:2002 Dell Computer Corp.
Bus 001 Device 005: ID 413c:1002 Dell Computer Corp. Keyboard Hub
Bus 001 Device 001: ID 0000:0000
```

Para verificar los recursos usados por el hardware podemos analizar los ficheros `interrupts`, `ioports` y `dma` del directorio `/proc`

- `/proc/interrupts`: muestra el número de interrupciones por IRQ (para x86)
- Ejemplo: sistema con una sola CPU

```
# cat /proc/interrupts
           CPU0
  0:   80448940          XT-PIC  timer
  1:     174412          XT-PIC  keyboard
  2:          0          XT-PIC  cascade
  8:          1          XT-PIC  rtc
 10:     410964          XT-PIC  eth0
 12:      60330          XT-PIC  PS/2 Mouse
 14:    1314121          XT-PIC  ide0
 15:    5195422          XT-PIC  ide1
NMI:          0
ERR:          0
```

- la primera columna muestra el número de IRQ, la segunda el numero de interrupciones por IRQ, la tercera el tipo de interrupción y la cuarta el dispositivo localizado en esa IRQ
- Definiciones

- XT-PIC: *XT-Programmable Interrupt Controller*, controlador de interrupciones de la arquitectura AT
- rtc: *Real Time Clock*
- cascade: para conectar dos PICs (8259A y 8259B)
- eth0: tarjeta Ethernet
- NMI (*Nonmaskable Interrup*), interrupción no-enmascarable

- Ejemplo: sistema con 2 CPUs (o 1 con hyperthreading)

```
# cat /proc/interrupts
           CPU0       CPU1
  0:   15126924          0    IO-APIC-edge  timer
  7:          2          0    IO-APIC-edge  parport0
  8:          0          0    IO-APIC-edge  rtc
  9:          0          0   IO-APIC-level  acpi
 14:     135534          1    IO-APIC-edge  ide0
169:      57807          0   IO-APIC-level  libata
177:     630479          0   IO-APIC-level  eth0
185:    1807688          0   IO-APIC-level  uhci_hcd, ehci_hcd
193:     154227          0   IO-APIC-level  uhci_hcd
201:          0          0   IO-APIC-level  uhci_hcd
209:    2153331          0   IO-APIC-level  uhci_hcd, Intel ICH
NMI:          0          0
ERR:          0
```

- Definiciones
  - IO-APIC (*I/O Advanced Programmable Interrupt Controller*): arquitectura de Intel para manejo de interrupciones en entorno multiprocesador (basado en el chip Intel 82093AA)
  - acpi (*Advanced Configuration and Power Interface*): interfaz estándar para configuración y manejo de energía gestionadas por el sistema operativo

- `/proc/ioports`: lista los puertos de entrada salida usados en el sistema

```
# cat /proc/ioports
0000-001f : dma1
0020-003f : pic1
0040-005f : timer
0060-006f : keyboard
```

```
0070-007f : rtc
0080-008f : dma page reg
00a0-00bf : pic2
00c0-00df : dma2
00f0-00ff : fpu
0170-0177 : ide1
01f0-01f7 : ide0
0376-0376 : ide1
03c0-03df : vga+
03f6-03f6 : ide0
03f8-03ff : serial(auto)
0cf8-0cff : PCI conf1
```

- `/proc/dma`: lista los canales ISA DMA registrados en uso

```
# cat /proc/dma
2: floppy
4: cascade
```

Discos duros

En arquitectura Intel nos vamos a encontrar normalmente con alguno de los siguientes tipos de discos:

1. SCSI
   - usuales en servidores de altas prestaciones (PCs, SPARC, etc.)
   - identificados en Linux como: `/dev/sda`, `/dev/sdb`,...
2. Serial ATA
   - Los más comunes
   - Linux los trata de forma similar a SCSI (`/dev/sda`,...)
     - soportados en el kernel 2.4.27 o superior (controlador libata)
3. IDE o Parallel ATA
   - Practicamente no se usan en la actualidad
   - Identificados en Linux como: `/dev/hda`, `/dev/hdb`, `/dev/hdc` y `/dev/hdd`
     - `hda`, `hdb` controlador IDE primario maestro y esclavo, respectivamente

    - `hdc`, `hdd` controlador IDE secundario maestro y esclavo, respectivamente

- Particiones: en Linux, las particiones en un disco se identifican con un número después del nombre del dispositivo:

    - podemos ver las particiones con el comando `fdisk -l` (sólo si superusuario):

```
# fdisk -l
Disco /dev/sda: 250.1 GB, 250059350016 bytes
255 cabezas, 63 sectores/pista, 30401 cilindros, 488397168 sectores en total
Unidades = sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico / físico): 512 bytes / 512 bytes
Tamaño E/S (mínimo/óptimo): 512 bytes / 512 bytes
Identificador del disco: 0x259d4594
Dispositivo Inicio    Comienzo      Fin      Bloques  Id  Sistema
/dev/sda1              63       80324       40131   de  Utilidad Dell
/dev/sda2         4179966   488396799   242108417    5  Extendida
/dev/sda5         4179968    64178175    29999104   83  Linux
/dev/sda6        64180224    68177919     1998848   82  Linux swap / Solaris
/dev/sda8        72179712   488396799   208108544   83  Linux
```

    - podemos ver las particiones montadas con el comando `df`:

```
# df
Sist. Fich                1K-bloques     Usado   Dispoñib Uso% Montado en
/dev/sda5                   29528148  20649776    7378420  74% /
udev                         1908232         4    1908228   1% /dev
tmpfs                         768136      1032     767104   1% /run
none                            5120         8       5112   1% /run/lock
none                         1920332      2756    1917576   1% /run/shm
cgroup                       1920332         0    1920332   0% /sys/fs/cg
/dev/sda8                  204842776 147789824   46647528  77% /home
```

    - Algunas opciones (para más opciones `man df`):
        - `-h`: muestra valores más fáciles de leer
        - `-i`: muestra información sobre inodos
        - `-T`: imprime el tipo de sistema de ficheros
        - `-l`: sólo muestra sistemas de ficheros locales

Dispositivos SCSI

Muy usados en sistemas de altas prestaciones (servidores)

- No sólo discos: cintas, CD-ROMs, escáneres, etc.
- Los dispositivos se conectan al bus en cadena (*daisy-chained*), actuando uno de ellos como controlador (interfaz con el host)

Evolución de SCSI

Versión	**Bus**	**Freq.**	**BW**	**Long.**	**N. disp.**
SCSI	8 bits	5 MHz	5 MB/s	6m	8
Fast SCSI	8 bits	10 MHz	10 MB/s	1.5-3m	8
Wide SCSI	16 bits	10 MHz	20 MB/s	1.5-3m	16
Ultra SCSI	8 bits	20 MHz	20 MB/s	1.5-3m	5-8
Ultra Wide SCSI	16 bits	20 MHz	40 MB/s	1.5-3m	5-8
Ultra2 SCSI	8 bits	40 MHz	40 MB/s	12m	8
Ultra2 Wide SCSI	16 bits	40 MHz	80 MB/s	12m	16
Ultra3 SCSI	16 bits	40 MHz DDR	160 MB/s	12m	16
Ultra-320 SCSI	16 bits	80 MHz DDR	320 MB/s	12m	16
Ultra-640 SCSI	16 bits	160 MHZ DDR	640 MB/s	12m	16

- Cada dispositivo en el bus (incluyendo el controlador) se identifica con un número (*SCSI address* o *target number*)
  - de 0 a 7 para bus de 8 bits y de 0 a 15 para bus de 16 bits
  - usualmente, el controlador tiene target 7 (en los dos buses)
- Algunos dispositivos, como RAID, tienen un sólo target y varios dispositivos lógicos:
  - LUN: *logical unit number*, identifica los dispositivos lógicos
  - en discos simples o cintas LUN=0

Ejemplo de configuración SCSI en Linux

Dispositivo	**Target**	**LUN**	**Disp. Linux**
Disco 0	0	-	`/dev/sda`
Disco 1	1	-	`/dev/sdb`
Cinta	5	-	`/dev/st0`
RAID disp. 0	6	0	`/dev/sdc`
RAID disp. 1	6	1	`/dev/sdd`
Controlador	7	-	-

Ejemplo, disco en Solaris:

- partición 6, del disco conectado al controlador 0, con target 9 y LUN 0:
  - `/dev/dsk/c0t9d0s6`

Otras versiones SCSI

- Serial Attached SCSI (SAS): bus serie, mayor velocidad (375-750 MB/s)
- iSCSI: Interner SCSI, permite el uso del protocolo SCSI sobre redes TCP/IP

2.5. Instalación de software

Tenemos, básicamente dos formas de instalar programas en Linux:

- Compilación e instalación desde las fuentes
  - Optimización para nuestro sistema
  - Más compleja
- Instalación desde paquetes precompilados
  - Menos optimización
  - Más sencilla

Instalación desde el código fuente

Pasos:

1. Descarga:
   - Normalmente se distribuye en forma de *tarballs*: ficheros `.tar.Z`, `.tar.gz`, `.tgz`, `.tar.bz2` o `.tbz`
2. Desempaquetado: comando `tar` (*Tape ARchive format*)
   - `tar` - crea y extrae ficheros de un archivo
   - Opciones principales:
     - `-c` o `--create` - Crea un archivo `tar`
     - `-t` o `--list` - Lista el contenido de un archivo
     - `-x` o `--extract` - Extrae los ficheros de un archivo
   - Otras opciones

- `-f` o `--file` *fich* - Usa el archivo *fich* (por defecto "-" que significa entrada/salida estándar)
- `-v` o `--verbose` - Lista los ficheros según se van procesando
- `-z` o `--gzip` - Comprime/descomprime ficheros `gzip`
- `-j` o `--bzip2` - Comprime/descomprime ficheros `bzip2`

- Ejemplos
  - Muestra el contenido de un tar.gz
    `$ tar tzvf archivo.tar.gz | more`
  - Extrae un fichero tar.bz2
    `$ tar xjvf archivo.tar.bz2`
  - Crea un tar.gz con los ficheros del directorio dir
    `$ tar czvf archivo.tar.gz dir/`

3. Leer el fichero INSTALL, INSTALAR o similar

4. Configuración

   - El código fuente desarrollado con ayuda de las herramientas GNU (*autoconf*) contienen un script `configure`, que se encarga de:
     - chequear el entorno de compilación
     - chequear las librerías necesarias
     - generar los `Makefiles` que nos permitirán compilar el código
   - Ejecución
     - `./configure` <*opciones*>
   - Para ver opciones: `./configure --help`
   - Ejemplo:
     - `./configure --prefix=/opt`
     - instala el programa en `/opt` en vez de en el directorio por defecto (normalmente `/usr/local`)

5. Compilación

   - El proceso de configuración genera ficheros `makefile` o `Makefile` en los directorios del código fuente
     - indican reglas (*rules*) que especifican como ejecutar ciertas tareas (*targets*) sobre el código: compilar, enlazar, crear páginas de manual, instalar
   - Funcionamiento:

- `make` (ejecuta el *target* por defecto, normalmente todo, menos instalar)
- `make all` (si no existe el *target* por defecto)
- `make clean` (borra ficheros objetos, ejecutables, etc)

6. Instalación
   - Si la compilación terminó con éxito, simplemente
     - `make install` (instala el programa ejecutable, librerías, páginas de manual)

Librerías compartidas Dos tipos de ejecutables:

1. Enlazados estáticamente (*statically linked*): son "completos"
2. Enlazados dinámicamente (*dynamically linked*): para ejecutarse necesitan librerías instaladas en el sistema
   - ocupan menos que los estáticos
   - librerías compartidas por varios programas

Para ver las librerías que un ejecutable necesita usar `ldd`:

```
# ldd /bin/ln
     libc.so.6 => /lib/tls/libc.so.6 (0xb7ea3000)
 /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0xb7fea000)
```

El cargador dinámico Se encarga de cargar los ejecutables con las librerías que necesitan

- en linux es `ld-linux.so.2`
- los directorios con librerías son (además de `/lib/` y `/usr/lib/`) los indicados en el fichero `/etc/ld.so.conf`
  - si modificamos ese fichero, debemos ejecutar el comando `ldconfig`, que regenera, a partir de los directorios indicados en `/etc/ld.so.conf`, el fichero `/etc/ld.so.cache`
  - para ver las librerías compartidas: `ldconfig -p |less`
  - si queremos que el cargador cargue las librerías de un directorio particular, antes de mirar los indicados en `ld.so.conf` usamos la variable de entorno `LD_LIBRARY_PATH`
    - `export LD_LIBRARY_PATH=/usr/lib/old:/opt/lib"`

Gestores de paquetes

En la mayoría de distribuciones Linux, es posible obtener los programas precompilados en formato de paquetes

- Ventajas:
  - Fáciles de instalar y desinstalar
  - Fáciles de actualizar
  - Fácil control de los programas instalados
- Inconvenientes
  - Binarios menos optimizados
  - Problemas de dependencias de paquetes
  - Problemas si la base de datos de paquetes se corrompe

Formatos de paquetes más populares

- Paquetes `DEB` (distribución Debian)
- Paquetes `RPM` (**R**edHat **P**ackage **M**anager, distribuciones Fedora, RedHat, Mandriva, etc.)

Gestión de paquetes en Debian

La distribución Debian incluye un elevado número de paquetes (más de 17.000)
Varias herramientas para el manejo de esos paquetes.

- `dpkg` - herramienta de bajo nivel, para gestionar directamente los paquetes DEB
- `apt-`*xxx* - herramientas APT, permiten gestionar los paquetes, descargándolos de varias fuentes (CDs, ftp, http)
- `dselect` - herramienta de administración de paquetes basada en menús (alto nivel)
- `tasksel` - interfaz para instalación de tareas (grupos de paquetes relacionados)
- `aptitude` - *front-end* de APT para usar en consola
- `synaptic` - *front-end* de APT para usar en entorno gráfico

- `alien` - permite convertir e instalar paquetes de otro tipo, p.e. RPMs

Para más información ver el capítulo Debian package management de la Debian Reference (v2)

dpkg Permite instalar, actualizar o desinstalar paquetes DEB
Los paquetes DEB contienen:

- Los binarios que se van a instalar
- Metadatos, con información sobre el paquete, *scripts* para su configuración, lista de dependencias, etc.

Nombre de los paquetes:

- *paquete_ versión-build_ arquitectura*.deb, donde
  - *paquete* - nombre de la aplicación
  - *versión* - número de versión de la aplicación
  - *build* - número de "compilación" (subversión)
  - *arquitectura* - plataforma para la que está compilado
- Ejemplo:
  - `ethereal_0.10.11-1_i386.deb`

Instalación y eliminación de paquetes con dpkg:

- Instalación de paquetes

  `dpkg --install` *paquete*`.deb`, o
  `dpkg -i` *paquete*`.deb`

  - la instalación chequea la existencia de dependencias, paquetes en conflicto, sobreescritura de ficheros existentes, etc.
  - se puede forzar la instalación usando la opción `--force-`*cosas*, donde *cosas*
    - `conflicts` - permite la instalación de paquetes en conflicto
    - `overwrite` - sobrescribe un fichero de un paquete con otro
    - `overwrite-dir` - sobrescribe el directorio de un paquete con uno nuevo
    - etc.
  - para ver todas las opciones de forzado hacer: `dpkg --force-help`

- Eliminación de paquetes, manteniendo los ficheros de configuración

  `dpkg --remove` *paquete*, o
  `dpkg -r` *paquete*

- Eliminación total de paquetes, eliminando los ficheros de configuración

  `dpkg --purge` *paquete*, o
  `dpkg -P` *paquete*

- Reconfiguración de un paquete ya instalado

  `dpkg-reconfigure` *paquete*

Información sobre los paquetes

- Listar paquetes

  `dpkg --list [`*patrón*`]`, o `dpkg -l [`*patrón*`]`

  - si no se pone *patrón* muestra los paquetes instalados
  - ejemplo

```
# dpkg -l 'telnet*'
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Nome                  Versión               Descripción
+++-=====================-=====================-======================================
ii  telnet                0.17-29               The telnet client
un  telnet-client        <ningunha>            (non hai ningunha descripción dispoñible)
un  telnet-hurd          <ningunha>            (non hai ningunha descripción dispoñible)
un  telnet-server        <ningunha>            (non hai ningunha descripción dispoñible)
pn  telnet-ssl           <ningunha>            (non hai ningunha descripción dispoñible)
pn  telnetd              <ningunha>            (non hai ningunha descripción dispoñible)
un  telnetd-hurd         <ningunha>            (non hai ningunha descripción dispoñible)
pn  telnetd-ssl          <ningunha>            (non hai ningunha descripción dispoñible)
```

- Las tres primeras columnas representan:
  - **Estado de selección**: indica el estado del paquete para su uso con dselect
    - u, *Unknown* - estado no conocido
    - i, *Install* - paquete seleccionado para instalar (se instala con `dselect install`)
    - r, *Remove* - paquete seleccionado para eliminar (se elimina con `dselect install`)
    - p, *Purge* - paquete seleccionado para purgar (se elimina con `dselect install`)
    - h, *Hold* - paquete retenido (no puede actualizarse)
  - **Estado actual**: indica el estado actual del paquete
    - n, *Not Installed* - paquete no instalado
    - i, *Installed* - paquete instalado en el sistema
    - c, *Config-files* - paquete no instalado, pero ficheros de configuración presentes (p.e. después de un remove)
    - u, *Unpacked* - paquete desempaquetado y listo para instalación
    - f, *Failed-config* - durante la instalación falló la configuración del paquete
    - h, *Half-installed* - paquete a medio instalar debido a algún problema
  - **Condiciones de error**
    - h, *Hold* - paquete retenido, no puede instalarse o eliminarse
    - r, *Reinstallation Required* - necesita reinstalarse
- Información y estado del paquete

- información general

  `dpkg --print-avail` *paquete*, o `dpkg -p` *paquete*

- estado del paquete

  `dpkg --status` *paquete*, o `dpkg -s` *paquete*

- ejemplo

```
# dpkg -s wget
Package: wget
Status: install ok installed
Priority: important
Section: web
Installed-Size: 1428
Maintainer: NoÃšl KÃ¶the <noel@debian.org>
Architecture: i386
Version: 1.10-2
Depends: libc6 (>= 2.3.2.ds1-21), libssl0.9.7
Conflicts: wget-ssl
Conffiles:
 /etc/wgetrc a9465704a21e403be628d38e10b0e128
Description: retrieves files from the web
 Wget is a network utility to retrieve files from the Web
  ....
```

- Ficheros de un paquete

  `dpkg --listfiles` *paquete*, o `dpkg -L` *paquete*

  - ejemplo

```
dpkg -L wget
/.
/etc
/etc/wgetrc
/usr
/usr/bin
/usr/bin/wget
/usr/share
/usr/share/doc
/usr/share/doc/wget
/usr/share/doc/wget/AUTHORS
/usr/share/doc/wget/ChangeLog.README
...
```

- Paquete al que pertenece un fichero

`dpkg --search` *fichero*, o `dpkg -S` *fichero*

- ejemplo

```
# dpkg --search /usr/bin/wget
wget: /usr/bin/wget
```

- Más información: ver ficheros en el directorio `/var/lib/dpkg/`
  - Fichero `/var/lib/dpkg/available`
    - muestra los paquetes disponibles
  - Fichero `/var/lib/dpkg/status`
    - muestra el estado de los paquetes
  - `dpkg` lee estos ficheros para obtener información sobre los paquetes

APT - Advanced Packaging Tools Permite descargar e instalar paquetes desde una fuente local y/o remota
Fuentes de apt: fichero `/etc/apt/sources.list`

```
# See sources.list(5) for more information
deb ftp://ftp.rediris.es/debian/ stable main contrib non-free
deb http://security.debian.org/ stable/updates main contrib non-free
#Para descargar fuentes, a través de apt-get source
deb-src ftp://ftp.rediris.es/debian/ stable main
```

- formato de `sources.list`

  `deb` *uri distribución componente* #Para binarios
  `deb-src` *uri distri. componente* #Para ficheros fuente

- *componente* puede ser:
  - `main` - conjunto principal de paquetes
  - `contrib` - paquetes adicionales
  - `non-free` - paquetes que no son libres

El fichero `sources.list` puede modificarse

- editándolo directamente, o
- a través del comando `apt-setup`

Opciones de configuración de `APT`

- Fichero `/etc/apt/apt.conf`
- Ficheros en el directorio `/etc/apt/apt.conf.d`

Cuando el fichero `sources.list` contiene referencias a más de una distribución (por ejemplo, estable y pruebas), APT asigna una prioridad a cada versión disponible

- es posible seleccionar una distribución ojetivo (*target release*) a la que se le asigna una mayor prioridad:
  - crear un fichero en el directorio `/etc/apt/apt.conf.d`, de nombre, por ejemplo, `99apt-default-release.conf` que contenga la línea

    `APT::Default-Release "`*`distribution`*`";`

    con *`distribution`* igual a *`stable`*, *`testing`* o *`unstable`*
- Si queremos instalar un paquete de una distribución distinta a la por defecto, añadir las líneas necesarias en el `sources.list` y usar `apt-get` con la opción `-t`

  `# apt-get -t `*`distribution`*` install `*`package`*
- podemos usar `apt-cache policy` para ver la política de prioridades configurada
- ver `man apt_preferences` y Debian package management para más detalles

Comando `apt-get`
Comando principal de las herramientas APT Permite descargar, instalar, actualizar o borrar un paquete

1. Actualizar la lista de paquetes

   `apt-get update`
2. Instalar un paquete

   `apt-get install` *nombre_paquete*
3. Actualizar los paquetes

   `apt-get upgrade`

- debe hacerse un `apt-get update` antes de un `apt-get upgrade`

4. Eliminar paquetes

   `apt-get remove` *nombre_paquete*

5. Actualizar la distribución

   `apt-get dist-upgrade`

   - maneja inteligentemente los cambios de dependencias debidos a nuevas versiones de paquetes

6. Eliminar los paquetes descargados

   - Cuando se instala un paquete a través de `apt-get` se guarda una copia en `/var/cache/apt/archives/`

     `apt-get clean` #Elimina todos los paquetes descargados
     `apt-get autoclean` #Elimina sólo los paquetes obsoletos

7. Descargar ficheros fuente

   `apt-get source` *nombre_paquete*

   - con la opción `--compile` compila el paquete después de descargarlo (y genera el `.deb`)

8. Descargar dependencias para compilar un paquete

   `apt-get build-dep` *nombre_paquete*

`apt-get` acepta diversas opciones, por ejemplo:

- `-s` - simula la acción, pero no instala nada
- `-y` - responde `y` a todas las preguntas

para más opciones `man apt-get`

Dependencias entre paquetes
Los paquetes pueden depender unos de otros:

- El paquete A depende (*Depends*) del paquete B si B es absolutamente necesario para usar A
- El paquete A recomienda (*Recommends*) el paquere B si se considera que la mayoría de los usuarios no querrían A sin las funcionalidades que proporciona B
- El paquete A sugiere (*Suggests*) el paquete B si B está relacionado y mejora las funcionalidades de A
- El paquete A está en conflicto (*Conflicts*) con B en el caso de que A no funcionae correctamente si B está instalado

Otras herramientas APT

1. `apt-cache` - permite manipular la caché de paquetes de APT, buscando paquetes o obteniendo información sobre los mismos
   - Ejemplo: buscar el paquete que contiene el firefox

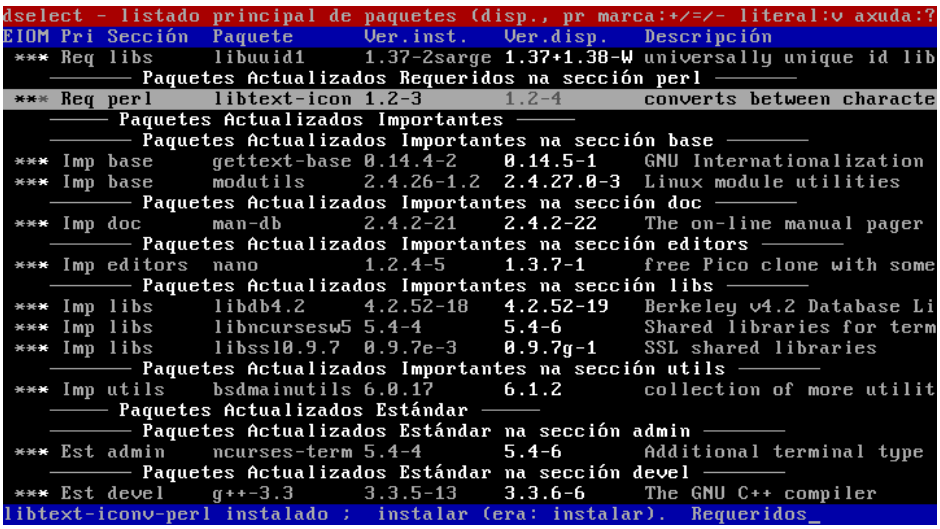
```
# apt-cache search firefox
bookmarkbridge - tool to synchronize bookmarks between browsers
gtkcookie - Editor for cookie files
latex-xft-fonts - Xft-compatible versions of some LaTeX fonts
libflash-mozplugin - GPL Flash (SWF) Library - Mozilla-compatible plugin
mozilla-firefox - lightweight web browser based on Mozilla
mozilla-firefox-dom-inspector - tool for inspecting the DOM of pages in Mozilla Fi
mozilla-firefox-gnome-support - Support for Gnome in Mozilla Firefox
mozilla-firefox-locale-af-za - Mozilla Firefox Afrikaans language/region package
...
```

   - el argumento puede ser una expresión regular

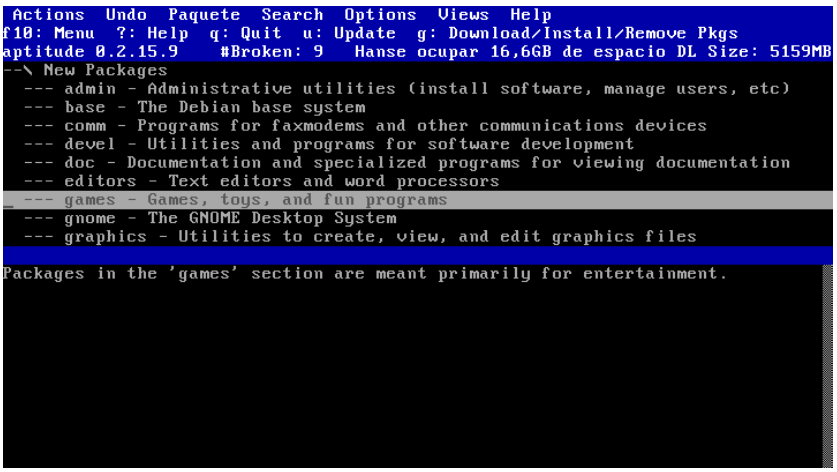
2. `apt-build` - permite descargar, compilar e instalar un paquete a partir de las fuentes

dselect, aptitude, tasksel, synaptic Interfaces del gestor de paquetes
Proporcionan interfaces para consola o gráficas para simplificar el manejo de los paquetes

- Ejemplo de `dselect`



- Ejemplo de `aptitude`



- `aptitude` tiene opciones similares a `apt-get`
  - `aptitude update`: actualiza la lista de paquetes
  - `aptitude search <nombre>`: busca paquetes
  - `aptitude show <nombre_paquete>`: muestra información del paquete
  - `aptitude install <nombre_paquete>`: instala paquetes
  - `aptitude remove <nombre_paquete>`: desinstala paquetes

- `aptitude purge <nombre_paquete>`: desinstala paquetes y sus archivos de configuración
- `aptitude clean`: elimina copias en cache de los ficheros deb descargados
- `aptitude autoclean`: elimina copias en cache de ficheros deb descargados obsoletos
- `aptitude hold <nombre_paquete>`: fuerza a que un paquete permanezca en su versión actual, y no se actualice
- `aptitude safe-upgrade`: actualiza los paquetes instalados, sin eliminar ninguno
- `aptitude full-upgrade`: actualiza los paquetes instalados, eliminando paquetes si es necesario

- `aptitude` podría manejar las dependencias mejor que `apt-get`, pero es menos estable

alien Convierte paquetes entre diferentes formatos
Soporta los formatos Red Hat rpm, Debian deb, Stampede slp, Slackware tgz, y Solaris pkg

- Por defecto, convierte paquetes a formato deb
- Algunas opciones (más opciones, ver página de manual):
  - `--to-rpm` o `-r` - crea un paquete rpm
  - `--to-tgz` o `-t` - crea un paquete tgz
  - `--to-slp` - crea un paquete slp
  - `--to-pkg` o `-p` - crea un paquete pkg
  - `--install` o `-i` - instala el paquete despues de crearlo
- Ejemplo:

```
# alien wget-1.9.1-5.i386.rpm
wget_1.9.1-6_i386.deb generated
```

Paquetes RPM: RedHat Package Manager

Instala software a partir de ficheros `.rpm`, manteniendo control de las dependencias

- Fichero RPM:

  nombre-versión-<release>.<arquitectura>.rpm

  - Ejemplos:

    `wget-1.9.1-5.i386.rpm`
    `xsnow-1.42-14.src.rpm`

  - Muchos RPMs pueden obtenerse en rpmfind.net
  - Otro repositorio: atrpms.net

- El sistema RPM mantiene una base de datos con información de los paquetes instalados en el sistema

  - si hay problemas, intentar reconstruirla con:

    `rpm --rebuilddb`

Comando `rpm` El comando `rpm` permite:

- instalar, actualizar y eliminar paquetes
- validar la integridad de un paquete
- consultar la base de datos RPM para obtener información
- construir un paquete binario a partir de las fuentes

Para más información sobre rpm:

- rpm Home Page
- RPM HOWTO
- Fedora Project Developer's Guide: Building RPM Packages

1. Instalar un rpm

   `rpm -i` *fichero.rpm*, o `rpm --install` *fichero.rpm*
   `rpm -ivh` *fichero.rpm* # Da una salida más visual

   - El proceso de instalación falla si detecta dependencias o si tiene que sobreescribir algún fichero existente
   - Opciones

- `--force` - Fuerza a `rpm` para que sobreescriba paquetes o ficheros existentes
- `--nodeps` - No chequea dependencias

- Ejemplo

```
# rpm -ivh xsnow-1.42-14.i386.rpm
Preparing...    ########################################
xsnow           ########################################
```

- En vez del fichero rpm puede usarse el URL del fichero, p.e.:

```
# rpm -ivh ftp://rpmfind.net/linux/fedora/core/2/i386/os/Fedora/RPMS/xs
Preparing...    ########################################
xsnow           ########################################
```

2. Borrar un paquete instalado

   `rpm -e` *paquete*, o `rpm --erase` *paquete*

   - Ejemplo

     ```
     # rpm -e xsnow
     ```

3. Actualizar un paquete

   `rpm -U` *fichero.rpm*, o `rpm --upgrade` *fichero.rpm*
   `rpm -F` *fichero.rpm*, o `rpm --freshen` *fichero.rpm*

   - si hay una versión del paquete instalada, la borra e instala la nueva; si no hay ninguna versión, simplemente instala la nueva
   - la opción `F` sólo actualiza si hay una versión más antigua instalada

4. Validar la integridad de un paquete

   `rpm --checksig` *fichero.rpm*, o `rpm -K` *fichero.rpm*

   - es necesario importar las claves públicas con el que se firmó el paquete
   - Ejemplo (buscamos la clave pública en el repositorio, por ejemplo, para paquetes de Fedora):

     ```
     # rpm -K xsnow-1.42-14.i386.rpm
     xsnow-1.42-14.i386.rpm: (SHA1) DSA sha1 md5 (GPG) NOT OK (MISSING KEYS:
     # rpm --import 4F2A6FD2.txt
     # rpm -K xsnow-1.42-14.i386.rpm
     xsnow-1.42-14.i386.rpm: (sha1) dsa sha1 md5 gpg OK
     ```

5. Información del paquete: uso `rpm -q` o `rpm --query`

   `rpm -q` [opciones] *paquete* # si el paquete está instalado
   `rpm -qp` [opciones] *fichero.rpm* # si el paquete **no** está instalado
   `rpm -qa` # muestra todos los paquetes instalados

   - Ejemplo:

     ```
     # rpm -qa |grep kernel
     kernel-smp-2.4.20-31.9
     kernel-pcmcia-cs-3.1.31-13
     kernel-source-2.4.20-31.9
     ```

   - Opciones de información

     a) Listar ficheros de un paquete
     `rpm -ql` *paquete*

     b) Determinar a que paquete pertenece un fichero
     `rpm -qf` *fichero*
     - Ejemplo:

       ```
       # rpm -qf /usr/bin/a2ps
       a2ps-4.13b-28
       ```

     c) Información del paquete
     `rpm -qi` *paquete*

     d) Requisitos previos (paquetes de los que depende)
     `rpm -qR` *paquete*

6. Verificar si algún fichero del paquete ha cambiado

   `rpm -V` *paquete*, o `rpm --verify` *paquete*

   - Ejemplo:

     ```
     # rpm -V pam
     S.5....T  c /etc/pam.d/system-auth
     ```

   - el fichero de configuración `system-auth` ha cambiado en tamaño (`S`), suma MD5 (`5`) y fecha de modificación (`T`)
   - otros indicadores:
     - `U`/`G` - cambio en el usuario/grupo del fichero
     - `M` - cambio en permisos o tipo de fichero

7. Compilar un paquete fuente

   - El paquete fuente se puede instalar con `rpm -i`

     ```
     rpm -ivh xsnow-1.42-14.src.rpm
     ```

   - los ficheros fuente se descomprimen en

     ```
     /usr/src/.../SOURCES/
     ```

   - `/usr/src/.../SPECS/` contiene el fichero de `spec`, que indica como compilar el código
   - el paquete se compila con el comando `rpmbuild`, generándose el RPM para instalar:

     ```
     rpmbuild -ba fichero.spec
     ```

   - podemos hacerlo directamente desde el rpm

     ```
     rpmbuild --rebuild fichero.rpm
     ```

YUM - Yellowdog Updater Modified Gestor de paquetes para sistemas basados en RPM

- Funcionalidad similar a APT
- Herramienta estándar en Fedora
- `up2date` puede usar YUM para actualizar el sistema

Algunas opciones:

- Busqueda de paquetes

  ```
  yum search nombre
  ```

- Instalación

  ```
  yum install nombre
  ```

- Actualización

  ```
  yum update nombre
  ```

Ficheros de configuración:

- Configuración base: `/etc/yum.conf`
- Repositorios: `/etc/yum.repos.d/`

APT con RPMs Es posible usar APT con RPMs:

- Instalar el paquete `apt`
- Configurar las fuentes:
  - pueden añadirse más fuentes en el `/etc/apt/sources.list`
  - Ejemplo (para Fedora):

```
# ATrpms for Fedora Core 1
# Possible sections: at-stable, at-good, at-testing, at-bleeding
rpm http://apt.atrpms.net fedora/2/en/i386 at-testing
#rpm-src http://apt.atrpms.net fedora/2/en/i386 at-testing
```

3. Uso de la línea de comandos

Veremos conceptos básicos para usar nuestro sistema desde la línea de comandos

- Un análisis más detallado se puede ver en

  www.ac.usc.es/docencia/ASRI/Tema_3html/node5.html

3.1. El interprete de comandos (shell)

El shell se inicia cuando accedemos a nuestra cuenta
Proporciona:

- un interprete de comandos
- un entorno de programación

El shell nos permite ejecutar:

- Comandos externos, por ejemplo: `ls`, `cat`, `mkdir`, etc.
  - son programas ajenos al shell
  - cuando se lanzan inician un nuevo proceso
  - se buscan en los directorios indicados en la variable PATH
- Comandos internos (*builtin commands*), por ejemplo: `cd`, `bg`, `alias`, `eval`, `exec`, `pwd`, etc.

    - se ejecutan en el mismo proceso del shell, sin lanzar un nuevo proceso
    - ver el manual del shell para más información (o para el shell bash: `man bash-builtins`, o el comando `help`)
- En bash: para saber si un comando es externo o interno usar el comando interno `type`:

```
$ type cd
cd is a shell builtin
$ type cat
cat is /bin/cat
```

Principales shells:

- **sh** o *Bourne **sh**ell*: shell por defecto en las primeras versiones de UNIX
- **bash** o ***B**ourne **a**gain **sh**ell*: versión mejorada de **sh**
    - desarrollada en el proyecto GNU
    - es el shell por defecto en Linux
- **csh** o ***C sh**ell*: desarrollada para UNIX BSD, su sintaxis se basa en la del lenguaje C
- **tcsh** o ***T**urbo **C sh**ell*: versión mejorada de **csh**
- **ksh** o ***K**orn **sh**ell*: basado en Bourne shell con características del C shell

Otros shells:

- **ash** o ***A**lmquist **sh**ell*: clon ligero de **sh** (en Linux Debian, **dash** o *Debian ash*)
- **fish** o **Friendly Interactive Shell**: shell amigable para sistemas UNIX
- **zsh** o ***Z sh**ell*: extensión mejorada de **sh**, incorporando características de otros shells como bash, ksh y tcsh
- **rc shell**: shell del sistema operativo *Plan 9* de los Bell Labs., (existe un *porting* de **rc** para UNIX)
- **es shell**: reimplementación del **rc shell** para sistemas UNIX; basado en programación funcional

Para ver las shells conocidas ver el fichero `/etc/shells`

- El shell por defecto para cada usuario se especifica en el fichero `/etc/passwd`
- Para ver la shell por defecto: `echo $SHELL`
- Para ver la shell actual: `ps | grep $$`
- Para cambiar de shell, ejecutar el comando correspondiente, p.e. `/bin/csh`
  - para volver al shell anterior `exit` o `Ctrl-D`
- Para cambiar la shell por defecto: `chsh`

3.2. La línea de comandos

El shell nos permite enviar comandos al sistema
Los comandos usualmente constan de 4 componentes.

- el nombre del comando (con la ruta absoluta, si no está en el PATH)
- opciones, usualmente precedidas por uno o dos guiones (-)
- argumentos (o parámetros)

Ejemplo: comando `ls` (lista ficheros y directorios)

`$ ls` (lista los archivos del directorio actual)
`$ ls -l` (lista los archivos en formato detallado)
`$ ls -la /tmp` (lista todos los archivos del directorio `/tmp`)

En algunos casos no es necesario usar guión con las opciones, ya que el comando espera por lo menos una:

```
$ tar cf miarchivo.tar arch1 arch2 arch3
```

Pueden indicarse varios argumentos, separados por espacios en blanco

```
$ echo hola amigo
```

- Comando $\longrightarrow$ `echo`
- Argumento 1 $\longrightarrow$ `hola`
- Argumento 2 $\longrightarrow$ `amigo`

Varios espacios en blanco se interpretan como uno solo

```
$ echo hola            amigo
```

Para que interprete todos los espacios usar comillas simples o dobles

```
$ echo 'hola            amigo'
```

- Comando $\longrightarrow$ `echo`
- Argumento 1 $\longrightarrow$ `hola            amigo`

3.3. Comandos básicos

- Busqueda de información: `man`, `info`, `help`, `whatis`, `apropos`
  - Proporcionan información sobre otros comandos
  - Más detalles en: www.ac.usc.es/docencia/ASRI/Tema_3html/node1.html
- Ficheros y directorios
  - `cp`, `mv`, `rm` - copia, mueve y borra ficheros
  - `cd`, `mkdir`, `rmdir` - accede, crea y borra directorios
- Manejo de ficheros de texto
  - `cat`, `more`/`less` - muestra el contenido de un fichero (`more` o `less` lo hacen página a página)
  - `vi`, `nano`, `emacs` - potentes editores de consola (una explicación de `vi` en www.ac.usc.es/docencia/ASRI/Tema_3html/node19.html)
- Otros comandos básicos
  - `su`, `sudo` - permiten ejecutar comandos cambiando los permisos del usuario, o como administrador
  - `alias` - Permiten crear alias de comandos complejos (para eliminarlos `unalias`)

    ```
    $ alias l='ls -la'
    ```

  - `history` - muestra una lista con los últimos comandos ejecutados y permite reejecutarlos
- Manejo del historial de comandos

Comando	Descripción
`<up-arrow>/<down-arrow>`	Comando anterior/posterior
`!!`	Último comando ejecutado
`!`*n*	*n*-ésimo comando del historial
`!-`*n*	*n* comandos hacia atrás
`!`*cadena*	Último comando ejecutado que empieza por cadena
`!?`*cadena*	Último comando ejecutado que contiene cadena
`^`*cadena1*`^`*cadena2*	Ejecuta el último comando cambiando *cadena1* por *cadena2*
`Ctrl-r`	Busca hacia atrás en el historial
`fc`	Permite ver, editar y reejecutar comandos del historial

3.4. Variables de shell

Uso de variables:

- control del entorno (*environment control*)
- programación shell

Dos tipos

- variables locales: visibles sólo desde el shell actual
- variables globales o de entorno: visibles en todos los shells

El comando `set` permite ver las variables definidas en nuestra shell

- El nombre de las variables debe:
  - empezar por una letra o _
  - seguida por cero o mas letras, números o _ (sin espacios en blanco)

Uso de las variables

- Asignar un valor: *nombre_ variable=valor*

```
$ un_numero=15
$ nombre="Pepe Pota"
```

- Acceder a las variables: *${nombre_ variable}* o *$nombre_ variable*

```
$ echo $nombre
Pepe Pota
```

- Eliminar una variable: *unset nombre_ variable*

```
$ unset nombre
$ echo ${nombre}mo
mo
```

- Variables de solo lectura: *readonly nombre_ variable*

```
$ readonly nombre
$ unset nombre
bash: unset: nombre: cannot unset: readonly variable
```

Variables de entorno

Cada shell se ejecuta en un *entorno* (*environment*)

- el entorno de ejecución especifica aspectos del funcionamiento del shell
- esto se consigue a través de la definición de variables de entorno (o variables globales)
- algunas variables son:

Nombre	Propósito
`HOME`	directorio base del usuario
`SHELL`	shell por defecto
`USERNAME`	el nombre de usuario
`PWD`	el directorio actual
`PATH`	el *path* para los ejecutables
`MANPATH`	el *path* para las páginas de manual
`PS1`/`PS2`	*prompts* primario y secundario
`LANG`	aspectos de localización geográfica e idioma
`LC_*`	aspectos particulares de loc. geográfica e idioma

- Para definir una nueva variable de entorno: `export`

```
$ nombre="Pepe Pota"         # Define una variable de shell
$ echo $nombre               # Usa la variable en el shell
Pepe Pota                    # padre
$ export nombre              # Exporta la variable
$ bash                       # Inicia un nuevo shell
$ echo Mi nombre es $nombre # Intenta usar la variable
Mi nombre es Pepe Pota       # del shell padre
$
```

- La variable exportada (variable de entorno) es visible en el shell hijo
  - el shell hijo crea una copia local de la variable y la usa
  - las modificaciones de esa copia no afectan al shell padre
- Para ver las variables de entorno definidas usar `env` o `printenv`

Más detalles sobre las variables del shell en

www.ac.usc.es/docencia/ASRI/Tema_3html/node11.html

3.5. Expansiones del shell

La sustitución de una variable por su valor se conoce como *expansión de parámetros*

```
$ A=Pepe
$ echo $A
Pepe
```

Otras expansiones

- Expansión de nombres de ficheros (*globbing*)
- Expansión de comandos
- Expansión de llaves
- Expansión de la tilde
- Expansión aritmética

Para más detalles sobre la expansión del shell mirar el manual de bash, sección `EXPANSION`

Expansión de nombres de ficheros

Los *comodines* (*wildcards*) permiten especificar múltiples ficheros al mismo tiempo:

`$ ls -l *html` # Lista los ficheros del directorio actual con terminación html

- también se conoce como *expansión de la shell* o *globbing*

- podemos ver como se hace la expansión poniendo `set -x` o `set -o xtrace`
  - `set +x` para no ver detalles
- podemos desactivar la expansión con `set -f` o `set -o noglob`

Lista de comodines

Carácter	Corresponde a
`*`	0 o más caracteres
`?`	1 carácter
`[ ]`	uno de los caracteres entre corchetes
`[! ]` o `[^ ]`	cualquier carácter que no esté entre corchetes

Los ficheros "ocultos" (que empiezan por `.`) no se expanden

- debemos poner el `.` de forma explícita

Nota importante: en **bash** el comportamiento de los rangos depende de la configuración de nuestro sistema, en particular, de la definición de la variable `LC_COLLATE`

- si `LC_COLLATE=C`, `[L-N]` implica `LMN` y `[l-n]` implica `lmn`
- en otro caso (p.e. si `LC_COLLATE=.es_ES.UTF-8"` o `"gl_ES@euro"`) entonces `[L-N]` implica `LmMnN` y `[l-n]` implica `lLmMn`

Para referirnos a mayúsculas o minúsculas podemos usar los siguientes patrones:

- `[[:lower:]]`: corresponde a un carácter en minúsculas
- `[[:upper:]]`: corresponde a un carácter en minúsculas
- `[[:alpha:]]`: corresponde a un carácter alfabético
- `[[:digit:]]`: corresponde a un número

Para más detalles: `man 7 glob`

Expansión de comandos

Permite que la salida de un comando reemplace el propio comando
Formato:

`$(comando)` o `` `comando` ``

Ejemplos:

```
$ echo date
date
$ echo `date`
Xov Xul 21 13:09:39 CEST 2005
$ echo líneas en fichero=$(wc -l fichero)
```

`wc -l` cuenta el número de líneas en el fichero; el comando se ejecuta y su salida se pasa al `echo`

Expansión de llaves

Permite generar strings arbitrarios

- no tiene para nada en cuenta los ficheros existentes en el directorio actual

```
$ echo a{d,c,b}e
ade ace abe
```

Expansión de la tilde

Expande la tilde como directorio HOME del usuario indicado

- si no se indica usuario, usa el usuario actual

```
cd ~              # Accedemos al nuestro HOME
cd ~root          # Accedemos al HOME de root
ls ~pepe/cosas/   # Vemos el contenido del directorio
                  cosas de pepe
```

Expansión aritmética

Permite evaluar expresiones aritméticas enteras

- se usa `$(( expresión ))` o `$[ expresión ]`
- `expresión` tiene una sintaxis similar a la del lenguaje C

- permite operadores como `++`, `+=`, `&&`,...

- También se puede usar `let`

```
$ let numero=(numero+1)/2  #usar " si se dejan espacios en blanco
```

- Ejemplos:

```
$ echo $(((4+11)/3))
5
$ numero=15
$ echo $((numero+3))
18
$ echo $numero
15
$ echo $((numero+=4))
19
$ echo $numero
19
$ numero=$(((numero+1)/2))
$ echo $numero
10
```

Eliminación del significado especial

bash permite eliminar el significado de los caracteres especiales, usando comillas simples, dobles o \

Carácter	Acción
'	el shell ignora todos los caracteres especiales contenidos entre un par de comillas simples
"	el shell ignora todos los caracteres especiales entre comillas dobles excepto $, `y \
\	el shell ignora el carácter especial que sigue a \

Ejemplos:

```
ls "/usr/bin/a*"
echo '$PATH'
echo "$PATH"
echo I\'m Pepe
```

3.6. Redirección de la entrada/salida

Es posible cambiar la fuente de la entrada o el destino de la salida de los comandos

- toda la E/S se hace a través de ficheros
- cada proceso tiene asociados 3 ficheros para la E/S

Nombre	Descriptor de fichero	Destino por defecto
entrada estándar (*stdin*)	0	teclado
salida estándar (*stdout*)	1	pantalla
error estándar (*stderr*)	2	pantalla

  - por defecto, un proceso toma su entrada de la entrada estándar, envía su salida a la salida estándar y los mensajes de error a la salida de error estándar

Ejemplo

```
$ ls /bin/bash /kaka
ls: /kaka: Non hai tal ficheiro ou directorio # Error
/bin/bash            # Salida estándar
$
```

Para cambiar la entrada/salida se usan los siguientes caracteres:

Carácter	**Resultado**
`comando < fichero`	Toma la entrada de `fichero`
`comando > fichero`	Envía la salida de `comando` a `fichero`; sobreescribe cualquier cosa de `fichero`
`comando 2> fichero`	Envía la salida de error de `comando` a `fichero` (el 2 puede ser reemplazado por otro descriptor de fichero)
`comando >> fichero`	Añade la salida de `comando` al final de `fichero`
`comando << etiqueta`	Toma la entrada para `comando` de las siguientes lineas, hasta una línea que tiene sólo `etiqueta`
`comando 2>&1`	Envía la salida de error a la salida estándar (el 1 y el 2 pueden ser reemplazado por otro descriptor de fichero, p.e. `1>&2`)
`comando &> fichero`	Envía la salida estándar y de error a `fichero`; equivale a `comando > fichero 2>&1`
`comando1 | comando2`	pasa la salida de `comando1` a la entrada de `comando2` (*pipe*)

Ejemplos

- `ls -l > lista.ficheros`
  Crea el fichero `lista.ficheros` conteniendo la salida de `ls -l`

- `ls -l /etc >> lista.ficheros`
  Añade a `lista.ficheros` el contenido del directorio `/etc`

- `cat < lista.ficheros | more`
  Muestra el contenido de `lista.ficheros` página a página (equivale a `more lista.ficheros`)

- `ls /kaka 2> /dev/null`
  Envía los mensajes de error al dispositivo nulo (a la *basura*)

- `> kk`
  Crea el fichero `kk` vacío

- `cat > entrada`
  Lee información del teclado, hasta que se teclea `Ctrl-D`; copia todo al fichero entrada

- `cat << END > entrada`
  Lee información del teclado, hasta que se introduce una línea con `END`; copia todo al fichero `entrada`

- `ls -l /bin/bash /kaka > salida 2> error`
  Redirige la salida estándar al fichero `salida` y la salida de error al fichero `error`

- `ls -l /bin/bash /kaka > salida.y.error 2>&1`
  Redirige la salida estándar y de error al fichero `salida.y.error`; el orden es importante:

  ```
  ls -l /bin/bash /kaka 2>&1 > salida.y.error
  ```

  no funciona, por qué?

- `ls -l /bin/bash /kaka &> salida.y.error`
  Igual que el anterior

- `cat /etc/passwd > /dev/tty2`
  Muestra el contenido de `/etc/passwd` en el terminal `tty2`

  - usar el comando `tty` para ver el nombre del terminal en el que estamos

Comandos útiles con pipes y redirecciones

1. `tee`
   - copia la entrada estándar a la salida estándar y también al fichero indicado como argumento:
     - `ls -l | tee lista.ficheros | less`
       Muestra la salida de `ls -l` página a página y la almacena en `lista.ficheros`
   - Opciones:
     - `-a`: no sobreescribe el fichero, añade al final
2. `xargs`
   - permite pasar un elevado número de argumentos a otros comandos
   - lee la entrada estándar, y ejecuta el comando uno o más veces, tomando como argumentos la entrada estándar (ignorando líneas en blanco)
   - Ejemplos:

     ```
     $ locate README | xargs cat | fmt -60 >\
     /home/pepe/readmes
     ```
     `locate` encuentra los ficheros `README`; mediante `xargs` los ficheros se envían a `cat` que muestra su contenido; este se formatea a 60 caracteres por fila con `fmt` y se envía al fichero `readmes`

     ```
     $ locate README | xargs -i cp {} /tmp/
     ```
     copia los `README` en el directorio `/tmp`; la opción `-i` permite que {} sea reemplazado por los nombres de los ficheros
3. `exec`
   - ejecuta un programa reemplazando el shell actual con el programa (es decir, al programa se le asigna el PID del shell, dejando el shell de existir)

     ```
     $ echo $$ #$$ indica el PID del shell actual
     4946
     $ exec sleep 20
     ```
     En otro terminal, ejecutamos
     ```
     $ ps a | grep 4946
     4946 pts/13   Ss+   0:00 sleep 20
     ```

- si no se especifica el programa, exec puede usarse para redireccionar las entradas y salidas
  - Redirecciona la salida estándar a el fichero `/tmp/salida`

    ```
    $ exec > /tmp/salida
    ```
  - Redirecciona el fichero `/tmp/entrada` como entrada estándar

    ```
    $ exec < /tmp/entrada
    ```

3.7. Orden de evaluación

Desde que introducimos un comando hasta que se ejecuta, el shell ejecuta los siguientes pasos, y en el siguiente orden:

1. Redirección E/S
2. Sustitución (expansión) de variables: reemplaza cada variable por su valor
3. Sustitución (expansión) de nombres de ficheros: sustituye los comodines por los nombres de ficheros

Si no se tiene en cuenta ese orden, pueden aparecer problemas:

```
$ star=\*
$ ls -d $star
cuatro  dos  tres  uno
$ pipe=\|
$ cat uno $pipe more
cat: |: Non hai tal ficheiro ou directorio
cat: more: Non hai tal ficheiro ou directorio
```

Comando `eval`

Evalúa la línea de comandos 2 veces:

- la primera hace todas las substituciones
- la segunda ejecuta el comando

Ejemplo:

```
$ pipe=\|
$ eval cat uno $pipe more
Este es el fichero uno
...
$
```

- En la primera pasada reemplaza `$pipe` por |
- En la segunda ejecuta el comando `cat uno | more`

3.8. Ficheros de inicialización de bash

Cuando se inicia bash se leen automáticamente distintos ficheros de inicialización

- En estos ficheros el usuario define variables de entorno, alias, el prompt, el path, etc.
- Los ficheros que se leen dependen de la forma de invocar bash

Formas de invocar bash:

1. Invocado como un *login shell* interactivo
   - cuando entramos en el sistema con login y password, usamos `su -`, o iniciamos bash con la opción `--login`
   - cuando se inicia, se leen los siguientes ficheros:
     a) `/etc/profile`
     b) el primero que exista de : `~/.bash_profile`, `~/.bash_login` o `~/.profile`
   - al dejar el shell se lee `~/.bash_logout`
2. Invocado como un *non-login shell* interactivo
   - cuando lo iniciamos sin opciones (bash), abrimos una nueva ventana de comandos (entramos sin login ni password), o usamos `su`
   - se leen los ficheros:
     a) `/etc/bash.bashrc`
     b) `~/.bashrc`[2]
   - al salir no se ejecuta nada
3. Invocado como un shell no interactivo
   - por ejemplo, cuando se lanza un script
   - en un shell no interactivo, la variable `$PS1` no está disponible
   - se lee el fichero definido en la variable `BASH_ENV`

[2]Usualmente, desde `.bash_profile` se invoca al `bashrc` de la siguiente forma:
`if [ -f ~/.bashrc ]; then . ~/.bashrc; fi`

4. Programación de scripts de administración

Un administrador de sistemas debe crear scripts para realizar tareas complejas

- La mayoría de los ficheros de configuración de Unix son ficheros ASCII
- Disponemos de potentes herramientas para manejar estos ficheros

Veremos

- Programación de scripts con bash
- Herramientas de manejo de ficheros de texto usando expresiones regulares
- Programación en Python
- Introducción a Perl y Ruby

4.1. Programación Shell-Script

Bash (y otros *shells*) permiten programar *scripts*:

Script o programa *shell*: fichero de texto conteniendo comandos externos e internos, que se ejecutan línea por línea

- El programa puede contener, además de comandos
  1. variables
  2. constructores lógicos (`if...then`, `AND`, `OR`, etc.) y lazos (`while`, `for`, etc.)
  3. funciones
  4. comentarios

Para saber más:

- *Advanced Bash-Scripting Guide*, Mendel Cooper, Última revisión Mayo 2005, www.tldp.org/guides.html
- *The Deep, Dark Secrets of Bash*, Ben Okopnik, Linux Gazette, okopnik.freeshell.org/articles/4.html
- *Introduction to Shell Scripting*, Ben Okopnik, okopnik.freeshell.org/writings.html

Más detalles en:

www.ac.usc.es/docencia/ASRI/Tema_3html/node34.html

Ejecución de un script

Los scripts deben empezar por el *número mágico* `#!` seguido del programa a usar para interpretar el script:

- `#!/bin/bash` script de bash
- `#!/bin/sh` script de shell
- `#!/usr/bin/perl` script de perl

Las forma usuales de ejecutar un script es:

- darle permiso de ejecución al fichero y ejecutarlo como un comando:

```
$ chmod +x helloworld
./helloworld
```

- ejecutar una shell poniendo como argumento el nombre del script (sólo necesita permiso de lectura)

```
$ bash helloworld
```

- ejecutarlo en la shell actual

```
$ . helloworld
```

  o bien:

```
$ source helloworld
```

Paso de parámetros

Es posible pasar parámetros a un scripts: los parámetros se recogen en las variables `$1` a `$9`

Variable	Uso
`$0`	el nombre del script
`$1` a `$9`	parámetros del 1 al 9
`${10}`, `${11}`,...	parámetros por encima del 10
`$#`	número de parámetros
`$*`, `$@`	todos los parámetros

Ejemplo:

```
$ cat parms1.sh
#!/bin/bash
VAL=$((${1:-0} + ${2:-0} + ${3:-0}))
echo $VAL
$ bash parms1.sh 2 3 5
10
$ bash parms1.sh 2 3
5
```

El comando `shift` desplaza los parámetros hacia la izquierda el número de posiciones indicado:

```
$ cat parms2.sh
#!/bin/bash
echo $#
echo $*
echo "$1 $2 $3 $4 $5 $6 $7 $8 $9 ${10} ${11}"
shift 9
echo $1 $2 $3
echo $#
echo $*
$ bash parms2.sh a b c d e f g h i j k l
12
a b c d e f g h i j k l
a b c d e f g h i j k
j k l
3
j k l
```

Entrada/salida

Es posible leer desde la entrada estándar o desde fichero usando `read` y redirecciones:

```
#!/bin/bash
echo -n "Introduce algo: "
read x
echo "Has escrito $x"
echo -n "Escribe 2 palabras: "
read x y
echo "Primera palabra $x; Segunda palabra $y"
```

Si queremos leer o escribir a un fichero utilizamos redirecciones:

```
echo $X > fichero
read X < fichero
```

Este último caso lee la primera línea de `fichero` y la guarda en la variable `X`

- Si queremos leer un fichero línea a línea podemos usar `while`:

```
#!/bin/bash
# FILE: linelist
# Usar: linelist filein fileout
# Lee el fichero pasado en filein y
# lo salva en fileout con las lineas numeradas
count=0
while read BUFFER
do
   count=$((++count))
   echo "$count $BUFFER"» $2
done < $1
```

  - el fichero de entrada se va leyendo línea a línea y almacenando en BUFFER
  - `count` cuenta las líneas que se van leyendo

- El uso de lazos para leer ficheros es bastante ineficiente
  - deberían evitarse (por ejemplo, usar `cat fichero`)

Ejemplo de lectura de fichero

```
#!/bin/bash
# Usa $IFS para dividir la línea que se está leyendo
# por defecto, la separación es "espacio"
echo "Lista de todos los usuarios:"
OIFS=$IFS # Salva el valor de IFS
IFS=: # /etc/passwd usa ":"para separar los campos
cat /etc/passwd |
while read name passwd uid gid fullname ignore
do
  echo "$name ($fullname)"
done
IFS=$OIFS # Recupera el $IFS original
```

- El fichero `/etc/passwd` se lee línea a línea
  - para cada línea, sus campos se almacenan en las variables que siguen a `read`
  - la separación entre campos la determina la variable `$IFS` (por defecto, espacio en blanco)

Redirecciones

Las redirecciones y pipes pueden usarse en otras estructuras de control

Ejemplo: lee las 2 primeras líneas de un fichero

```
if true
then
  read x
  read y
fi < fichero1
```

Ejemplo: lee líneas de teclado y guardalas en un fichero temporal convirtiendo minúsculas en mayúsculas

```
#/bin/bash
read buf
while [ "$buf" ]
do
  echo $buf
  read buf
done | tr 'a-z' 'A-Z' > tmp.$$
```

Tests

Los comandos que se ejecutan en un shell tienen un *código* de salida, que se almacena en la variable `$?`

- si `$?` es 0 el comando terminó bien
- si `$?` es > 0 el comando terminó mal

Ejemplo:

```
$ ls /bin/ls
/bin/ls
$ echo $?
```

```
0
$ ls /bin/ll
ls: /bin/ll: Non hai tal ficheiro ou directorio
$ echo $?
1
```

Podemos chequear la salida de dos comandos mediante los operadores `&&` (AND) y `||` (OR)

- estos operadores actúan en cortocircuito:

  comando1 `&&` *comando2*
  comando2 sólo se ejecuta si *comando1* acaba bien
  comando1 `||` *comando2*
  comando2 sólo se ejecuta si *comando1* falla

- comandos `true` y `false`: devuelven 0 y 1, respectivamente

Ejemplo con `&&`:

```
$ ls /bin/ls && ls /bin/ll
/bin/ls
ls: /bin/ll: Non hai tal ficheiro ou directorio
$ echo $?
1
$ ls /bin/ll && ls /bin/ls
ls: /bin/ll: Non hai tal ficheiro ou directorio
$ echo $?
1
```

Ejemplo con `||`:

```
$ ls /bin/ls || ls /bin/ll
/bin/ls
$ echo $?
0
$ ls /bin/ll || ls /bin/ls
ls: /bin/ll: Non hai tal ficheiro ou directorio
/bin/ls
$ echo $?
0
```

Estructura `if...then...else`

Podemos usar el estado de salida de uno o varios comandos para tomar decisiones:

```
if comando1
then
  ejecuta otros comandos
elif comando2
then
  ejecuta otros comandos
else
  ejecuta otros comandos
fi
```

- debe respetarse la colocación de los `then`, `else` y `fi`
  - también puede escribirse `if` *comando1* `; then`
- el `elif` y el `else` son opcionales, no así el `fi`

Ejemplo:

```
$ cat if.sh
#!/bin/bash
if (ls /bin/ls && ls /bin/ll) >/dev/null 2>&1
then
  echo "Encontrados ls y ll"
else
  echo "Falta uno de los ficheros"
fi
$ bash if.sh
Falta uno de los ficheros
```

Comando `test`

Notar que `if` sólo chequea el código de salida de un comando, no puede usarse para comparar valores: para eso se usa el comando `test`
El comando test permite:

- chequear la longitud de un string
- comparar dos strings o dos números

- chequear el tipo de un fichero
- chequear los permisos de un fichero
- combinar condiciones juntas

`test` puede usarse de dos formas:

`test` *expresión*

o bien

`[` *expresión* `]`[3]

Si la expresión es correcta `test` devuelve un código de salida 0, si es falsa, devuelve 1:

- este código puede usarse para tomar decisiones:

```
if [ "$1" = "hola" ]
then
  echo "Hola a ti también"
else
  echo "No te digo hola"
fi
if [ $2 ]
then
  echo "El segundo parámetro es $2"
else
  echo "No hay segundo parámetro"
fi
```

- en el segundo `if` la expresión es correcta si `$2` tiene algún valor; falsa si la variable no está definida o contiene null (`""`)

Expresiones

Existen expresiones para chequear strings, números o ficheros

[3] Notar los espacios en blanco entre los [ ] y *expresión*

Chequeo de strings

Expresión	Verdadero sí
string	el string es no nulo ("")
-z *string*	la longitud del string es 0
-n *string*	la longitud del string no es 0
string1 = *string2*	los strings son iguales
string1 != *string2*	los strings son distintos

Chequeo de enteros

Expresión	Verdadero sí
int1 -eq *int2*	los enteros son iguales
int1 -ne *int2*	los enteros son distintos
int1 -gt *int2*	*int1* mayor que *int2*
int1 -ge *int2*	*int1* mayor o igual que *int2*
int1 -lt *int2*	*int1* menor que *int2*
int1 -le *int2*	*int1* menor o igual que *int2*

Chequeo de ficheros

Expresión	Verdadero sí
-e *file*	*file* existe
-r *file*	*file* existe y es legible
-w *file*	*file* existe y se puede escribir
-x *file*	*file* existe y es ejecutable
-f *file*	*file* existe y es de tipo regular
-d *file*	*file* existe y es un directorio
-c *file*	*file* existe y es un dispositivo de caracteres
-b *file*	*file* existe y es un dispositivo de bloques
-p *file*	*file* existe y es un pipe
-S *file*	*file* existe y es un socket
-L *file*	*file* existe y es un enlace simbólico
-u *file*	*file* existe y es *setuid*
-g *file*	*file* existe y es *setgid*
-k *file*	*file* existe y tiene activo el *sticky bit*
-s *file*	*file* existe y tiene tamaño mayor que 0

Operadores lógicos con `test`

Expresión	Propósito
`!`	invierte el resultado de una expresión
`-a`	operador AND
`-o`	operador OR
`( expr )`	agrupación de expresiones; los paréntesis tienen un significado especial para el shell, por lo que hay que *escaparlos*

Ejemplos:

```
$ test -f /bin/ls -a -f /bin/ll ; echo $?
1
$ test -c /dev/null ; echo $?
0
$ [ -s /dev/null ] ; echo $?
1
$ [ ! -w /etc/passwd ] && echo "No puedo escribir"
No puedo escribir
$ [ $$ -gt 0 -a \( $$ -lt 5000 -o -w file \) ]
```

Comando de test extendido A partir de la versión 2.02 de Bash se introduce el *extended test command*: `[[ expr ]]`

- permite realizar comparaciones de un modo similar al de lenguajes estándar:
  - permite usar los operadores && y || para unir expresiones
  - no necesita *escapar* los paréntesis

Ejemplos:

```
$ [[ -f /bin/ls && -f /bin/ll ]] ; echo $?
1
$ [[ $$ -gt 0 && ($$ -lt 5000 || -w file) ]]
```

Control de flujo

Además del `if` bash permite otras estructuras de control de flujo: `case`, `for`, `while` y `until`

Estructura `case`

```
case valor in
  patrón_1)
    comandos si value = patrón_1
    comandos si value = patrón_1 ;;
  patrón_2)
    comandos si value = patrón_2 ;;
  *)
    comandos por defecto ;;
esac
```

- si *valor* no coincide con ningún patrón se ejecutan los comandos después del `*)`
  - esta entrada es opcional
- *patrón* puede incluir comodines y usar el símbolo | como operador OR

Ejemplo:

```
#!/bin/bash
echo -n "Respuesta:" read RESPUESTA
case $RESPUESTA in
  S* | s*)
    RESPUESTA="SI";;
  N* | n*)
    RESPUESTA="NO ";;
  *)
    RESPUESTA="PUEDE";;
esac
echo $RESPUESTA
```

Lazos `for`

```
for var in lista
do
  comandos
done
```

- *var* toma los valores de la lista
  - puede usarse *globbing* para recorrer los ficheros

Ejemplo: recorrer una lista

```
LISTA="10 9 8 7 6 5 4 3 2 1"
for var in $LISTA
do
  echo $var
done
```

Ejemplo: recorrer los ficheros `*.bak` de un directorio

```
dir="/var/tmp"
for file in $dir/*.bak
do
  rm -f $file
done
```

Sintaxis alternativa, similar a la de C

```
LIMIT=10
for ((a=1, b=LIMIT; a <= LIMIT; a++, b--))
do
  echo "$a-$b"
done
```

Bucle `while`

```
while comando
do
  comandos
done
```

- se ejecuta mientras que el código de salida de `comando` sea cierto

Ejemplo:

```
while [ $1 ]
do
  echo $1
  shift
done
```

Bucle `until`

```
until comando
do
  comandos
done
```

- se ejecuta hasta que el código de salida de `comando` sea hace cierto

Ejemplo:

```
until [ "$1" = ""]
do
  echo $1
  shift
done
```

`break` y `continue` Permiten salir de un lazo (`break`) o saltar a la siguiente iteración (`continue`)

- `break` permite especificar el número de lazos de los que queremos salir (`break` *n*)

Ejemplo con `break`:

```
# Imprime el contenido de los ficheros hasta que
# encuentra una línea en blanco
for file in $*
do
  while read buf
  do
    if [ -z "$buf"]
    then
      break 2
    fi
    echo $buf
  done <$file
done
```

Ejemplo con `continue`:

```
# Muestra un fichero pero no las líneas de más
# de 80 caracteres
```

```
while read buf
do
  cuenta=`echo $buf | wc -c`
  if [ $cuenta -gt 80 ]
  then
    continue
  fi
  echo $buf
done <$1
```

Funciones

Podemos definir funciones en un script de shell:

```
funcion() {
  comandos
}
```

y para llamarla:

```
funcion p1 p2 p3
```

Siempre tenemos que definir la función antes de llamarla:

```
#!/bin/bash
# Definición de funciones
funcion1() {
  comandos
}
funcion2() {
  comandos
}
# Programa principal
funcion1 p1 p2 p3
```

Paso de parámetros La función referencia los parámetros pasados por posición, es decir, `$1`, `$2`, ..., y `$*` para la lista completa:

```
$ cat funcion1.sh
#!/bin/bash
funcion1()
{
  echo "Parámetros pasados a la función: $*"
```

```
  echo "Parámetro 1: $1"
  echo "Parámetro 2: $2"
}
# Programa principal
funcion1 "hola" "que tal estás" adios
$
$ bash funcion1.sh
Parámetros pasados a la función: hola que tal estás adios
Parámetro 1: hola
Parámetro 2: que tal estás
```

`return` Después de llamar a una función, `$?` tiene el código se salida del último comando ejecutado:

- podemos ponerlo de forma explícita usando `return`

```
#!/bin/bash
funcion2() {
  if [ -f /bin/ls -a -f /bin/ln ]; then
    return 0
  else
    return 1
  fi
}
# Programa principal
if funcion2; then
  echo "Los dos ficheros existen"
else
  echo "Falta uno de los ficheros - adiós"
  exit 1
fi
```

Otros comandos

`wait` Permite esperar a que un proceso lanzado en *background* termine

```
sort $largefile > $newfile &
ejecuta comandos
wait
usa $newfile
```

Si lanzamos varios procesos en background podemos usar `$!`

- `$!` devuelve el `PID` del último proceso lanzado

```
sort $largefile1 > $newfile1 &
SortPID1=$!
sort $largefile2 > $newfile2 &
SortPID2=$!
ejecuta comandos
wait $SortPID1
usa $newfile1
wait $SortPID2
usa $newfile2
```

`trap` Permite *atrapar* las señales del sistema operativo

- permite hacer que el programa termine limpiamente (p.e. borrando ficheros temporales, etc.) aún en el evento de un error

```
$ cat trap.sh
#!/bin/bash
cachado() {
  echo "Me has matado!!!"
  kill -15 $$
}
trap "cachado" 2 3
while true; do
  true
done
$ bash trap.sh
(Ctrl-C)
Me has matado!!!
Terminado
```

Las señales más comunes para usar con `trap` son:

Señal	Significado
0	salida del shell (por cualquier razón, incluido fin de fichero)
1	colgar
2	interrupción (Ctrl-C)
3	quit
9	kill (no puede ser parada ni ignorada)
15	terminate; señal por defecto generada por kill

`exit` Finaliza el script

- se le puede dar un argumento numérico que toma como estado de salida, p.e. `exit 0` si el script acaba bien y `exit 1` en caso contrario
- si no se usa `exit`, el estado de salida del script es el del último comando ejecutado

Referencias indirectas

Permiten definir variables cuyo contenido es el nombre de otra variable:

```
a=letra
letra=z
# Referencia directa
echo "a = $a"  # a = letra
# Referencia indirecta
eval a=\$$a
echo "Ahora a = $a"  # Ahora a = z
```

Las versiones de bash a partir de la 2 permiten una forma más simple para las referencias indirectas:

```
a=letra
letra=z
# Referencia directa
echo "a = $a"  # a = letra
# Referencia indirecta
echo "Ahora a = ${!a}"  # Ahora a = z
```

Otro ejemplo con `eval`

```
$ cat dni.sh
#!/bin/bash
dniPepe=23456789
dniPaco=98765431
echo -n "Nombre: "; read nombre
eval echo "DNI = \$dni${nombre}"
$ bash dni.sh
Nombre: Pepe
DNI = 23456789
```

Optimización de scripts

El shell no es especialmente eficiente a la hora de ejecutar trabajos pesados

- Ejemplo: script que cuenta las líneas de un fichero:

```
$ cat cuentalineas1.sh
#!/bin/bash
count=0
while read line
do
  count=$(expr $count + 1)
done < $1
echo "El fichero $1 tiene $count líneas"
```

  - si medimos el tiempo que tarda

```
$ time bash cuentalineas1.sh Quijote.txt
El fichero Quijote.txt tiene 36855 líneas
real 0m59.757s
user 0m17.868s
sys  0m41.462s
```

- Podemos mejorarlo si usamos aritmética de shell en vez de el comando `expr`

```
$ cat cuentalineas2.sh
#!/bin/bash
count=0
while read line
do
  count=$(($count+1))
done < $1
echo "El fichero $1 tiene $count líneas"
```

  - el tiempo ahora

```
$ time bash cuentalineas2.sh Quijote.txt
El fichero Quijote.txt tiene 36855 líneas
real 0m1.014s
user 0m0.887s
sys  0m0.108s
```

- Y todavía mejor:

```
$ cat cuentalineas3.sh
#!/bin/bash
count=$(wc -l $1 | cut -d " " -f 1)
echo "El fichero $1 tiene $count líneas"
$
$ time bash cuentalineas3.sh Quijote.txt
El fichero Quijote.txt tiene 36855 líneas
real 0m0.096s
user 0m0.005s
sys  0m0.009s
```

- Conclusiones
  - Intenta reducir el número de procesos creados al ejecutar el script, por ejemplo, usando las funciones aritméticas del shell
  - Siempre que sea posible, intenta usar comandos del shell (`wc`, `tr`, `grep`, `sed`, etc.) en vez de lazos

Depuración

Para depurar un script de shell podemos usar la opción `-x` o `-o xtrace` de bash:

- muestra en la salida estándar trazas de cada comando y sus argumentos, después de que el comando se haya expandido pero antes de que se sea ejecutado

```
$ bash -x cuentalineas3.sh Quijote.txt
++ wc -l Quijote.txt
++ cut -d ' ' -f 1
+ count=36855
+ echo 'El fichero Quijote.txt tiene 36855 líneas'
El fichero Quijote.txt tiene 36855 líneas
```

Es posible depurar sólo parte de un script:

- poner `set -x` o `set -xv` al inicio del trozo a depurar
- `set +x` o `set +xv` para cancelar

```
$ cat cuentalineas3.sh
#!/bin/bash
set -x
count=$(wc -l $1 | cut -d " "-f 1)
set +x
echo "El fichero $1 tiene $count líneas"
$
$ bash cuentalineas3.sh Quijote.txt
++ wc -l Quijote.txt
++ cut -d ' '-f 1
+ count=36855
+ set +x
El fichero Quijote.txt tiene 36855 líneas
```

5. Manejo de ficheros de texto

Los ficheros de configuración y logs de Unix son, normalmente, ficheros de texto

- se necesitan herramientas para manejar estos ficheros
- Unix dispone de potentes herramientas que hacen uso extensivo de expresiones regulares

5.1. Expresiones regulares

Muchos comandos de procesamiento y búsqueda de texto como `ed`, `grep`, `egrep`, `sed`, `awk` o `vi` usan expresiones regulares:

- permiten reconocer una serie de cadenas de caracteres que obedecen a cierto patrón
- Ejemplos
  - `egrep unix tmp.txt`
    busca en el fichero `tmp.txt` las líneas que contienen la palabra `unix`
  - `egrep '[Uu]nix' tmp.txt`
    busca las líneas que contienen `unix` o `Unix`
  - `egrep 'hel.' tmp.txt`
    busca las líneas que contienen `hel` seguido de cualquier carácter

- `egrep 'ab*c' tmp.txt`
  localiza las cadenas que empiecen por `a`, que continúen con 0 o más `b`, y que sigan con una `c`, por ejemplo: `abbbc` o `aaacb`, pero no `axc` o `cba`
- `egrep 't[^aeiouAEIOU][a-zA-Z]*' tmp.txt`
  localiza las cadenas que empiecen por `t`, seguido de algún carácter no vocálico y 0 o más apariciones de otro carácter

Importante: no debemos confundir las expresiones regulares con la sustitución de nombres de ficheros (*globbing*)

- si ponemos el último ejemplo sin comillas

  ```
  egrep t[^aeiouAEIOU][a-zA-Z]* tmp.txt
  ```

  la shell extiende los comodines y convierte este comando en:

  ```
  egrep tmp.txt tmp.txt
  ```

- para evitar esto, `siempre` usar comillas con las expresiones regulares

Comandos grep y sed

`grep` y `sed` son dos comandos que usan REGEXP

`grep` Busca en ficheros por un patrón determinado

`grep [`*`opciones`*`] `*`patrón`*` [`*`fichero...`*`]`

Opciones:

- `-E` o `egrep`: usa expresiones regulares extendidas
- `-F` o `fgrep`: interpreta los patrones no como expresiones regulares sino como cadenas de caracteres fijas
- `-R` o `rgrep`: lee todos los ficheros bajo cada directorio, recursivamente
- `-i` o `--ignore-case`: busca ignorando diferencias entre mayúsculas y minúsculas
- `-w` o `--word-regexp`: para forzar que la cadena reconocida sea una palabra completa
- `-l` o `--files-with-matches`: no muestra el contenido de la linea encontrada pero si que muestra el fichero que contiene la cadena buscada

- `-n` o `--line-number`: muestra el número de línea dentro del fichero
- `-v` o `--invert-match`: en lugar de sacar la lineas que cumplen la búsqueda sacará las que no cumplen

Si no especificamos fichero, `grep` usa la entrada estándar:

- podemos usarlo para probar las expresiones regulares:

```
$ egrep '[Uu]nix'
unix
unix
Unix
Unix
Linux
```

`sed` (***stream editor***) Editor de flujo; permite realizar transformaciones básicas de un flujo de entrada (un fichero o una entrada desde una tubería)

Formato (para substituciones):

`sed` [*opciones*] 's/*REGEXP*/*reemplazo*/*flag*' [*fichero*]

Algunos comandos:

- `s` substitución
- `d` borrado
- `i\`, `a\`, añade antes/después de la línea afectada
- `c\` reemplaza la línea afectada

Algunas ociones:

- `-e` *comando*: añade *comando*
- `-i` edita el fichero *in-place*
- `-n` suprime la salida

Algunos flags:

- `g`: aplica los cambios globalmente (por defecto, sólo se cambia la primera aparición en cada línea)

- `p` imprime las líneas afectadas, incluso con la opción `-n`.
- *NUMERO*: reemplaza la aparición número *NUMERO*
- `w` *fichero*: escribe las líneas con sustituciones al fichero indicado

Ejemplo: cambia, en el fichero `amigos`, todas las apariciones de `pepe` y `paco` por `Pepe` y `Paco`, respectivamente:

> `$ sed -e 's/pepe/Pepe/g' -e 's/paco/Paco/g' amigos` (también `sed 's/pepe/Pepe/g ; s/paco/Paco/g' amigos`)

Ejemplo: cambia `pepe` por `Pepe`, pero sólo en las líneas que tengan `Potamo`

```
$ sed '/Potamo/s/pepe/Pepe/g' amigos
```

Ejemplo: muestra sólo las lineas que contengan `jaime`

```
$ sed -n '/jaime/p' amigos
```

Ejemplo: borra las lineas que contengan `jaime`

```
$ sed '/jaime/d' amigos
```

Ejemplo: cambia las lineas que contengan `jaime` por otra cosa

```
$ sed '/jaime/c\BORRADO' amigos
```

Ejemplo: inserta una línea, con la palabra 'APARICION', antes de las líneas que contengan `jaime`

```
$ sed '/jaime/i\APARICION' amigos
```

Ejemplo: reemplaza, en cada línea de `fichero`, la quinta ocurrencia de `stop` por `STOP`

```
$ sed 's/stop/STOP/5' fichero
```

Ejemplo: igual que antes pero guarda cada línea reemplazada en el fichero `f2`

```
$ sed 's/stop/STOP/5w f2' fichero
```

Indicación de líneas: podemos especificar las líneas del fichero en las que queremos que se realicen las operaciones:

> `sed '3s/stop/STOP/g'` (reemplaza sólo en la línea 3)
> `sed '3,10s/stop/STOP/g'` (reemplaza de la línea 3 a la 10)
> `sed '3,$s/stop/STOP/g'` (reemplaza de la línea 3 al final)
> `sed '!3s/stop/STOP/g'` (reemplaza en todas las líneas menos la 3)

Operador `&`: se sustituye por el patrón reconocido

Ejemplo: reemplaza `stop` por `<stop>`

```
$ sed '3s/stop/<&>/g' fichero
```

Comandos desde fichero: la opción -f permite leer comandos de `sed` agrupados en un fichero

Ejemplo: reemplazo desde la línea 1 hasta una línea que comience por `END` (o el final, si no hay ninguna)

```
$ cat file.sed
1,/^END/{
  s/[Ll]inux/GNU\/Linux/g
  s/samba/Samba/g
}
$ sed -f file.sed fichero
```

Más información: `sed` es un comando muy complejo con muchas posibilidades

Para saber más:

- mirar la página de `info` de `sed`
- Sed - An Introduction
- Ejemplos con `sed`
- Sed by example, IBM developerworks
- sed & awk, by Dale Dougherty, Arnold Robbins, O'Reilly

o, simplemente, busca *sed tutorial* en *google*

Expresiones regulares básicas

UNIX admite dos tipos de expresiones regulares: básicas y extendidas

- las básicas son las clásicas de UNIX, aunque se consideran obsoletas en POSIX
- aplicaciones como `grep` o `sed` las usan por defecto

- para usar las extendidas:
  - `grep` ⟶ `egrep` o `grep -E`
  - `sed` ⟶ `sed -r`
- las expresiones extendidas proporcionan más potencia

La mayoría de los caracteres son tratados como literales:

- concuerdan (*match*) consigo mismos:
  - `a` concuerda con `a`, `ab` con `ab`, etc.
- la excepción son los metacaracteres:

```
. [ ] ^ $ * ( ) \
```

ER de un sólo carácter

ER	concuerda con
`.`	cualquier carácter
`[ ]`	cualquiera de los caracteres entre corchetes, p.e. `[abc]` concuerda con `a`, `b` o `c`; `[a-z]` concuerda con cualquier letra minúscula
`[^ ]`	cualquier carácter que no esté entre corchetes
`^`	principio de línea
`$`	final de línea
`*`	0 o más ocurrencias de la expresión regular anterior
`\( \)`	permite agrupar ER
`\`	*escapa* un metacarácter

- Dentro de `[ ]` los metacaracteres pierden su significado especial: p.e. `[a.]c` concuerda con `ac` y `.c`
- Para incluir un carácter `]` en una lista colocarlo al principio; para incluir un `^` en cualquier lugar menos al principio; para incluir un `-` al final: p.e. `[a^]c` concuerda con `ac` y `^c`

Ejemplos:

ER	concuerda con
`a..c`	cadena que empiece por `a`, seguida por dos caracteres y `c`: `a00c`, `xaxxcxx`, `aacc`,...
`0[abc]0`	cadenas que tengan un `0` seguido de un carácter `a`, `b`, o `c` y seguido de otro `0`: `0a0`, `00ab0b0`, `bc0c0`,...
`0[^abc]0`	cadenas que tengan un `0` seguido de un carácter distinto a `a`, `b`, o `c` y seguido de otro `0`
`0[a-z]0`	cadenas que tengan un `0` seguido de una letra minúscula, y `0`
`^abc`	líneas que empiecen por `abc`
`abc$`	líneas que terminen por `abc`
`ab*c`	cadenas que empiecen por `a`, que continúen con 0 o más `b`, y una `c`: `abc`, `ac`, `abbc`, `aaccab`,... pero no `cba` o `aaab`
`b[cq]*e`	cadenas que empiecen por `b`, que continúen con 0 o más `c` o `q`, y una `e`: `be`, `bcce`, `bccqqee` o `bqqqce`
`.*`	cualquier cadena
`abc.*`	cualquier cadena que empiece por `abc`
`0\(abc\)*0`	cadenas que tengan un `0` seguido de 0 o más ocurrencias de `abc`, y seguido de otro `0`: `0abc0`, `00`, `0abcabc0`,..., pero no `0ac0` o `0cba0`
`^#.*\.$`	línea que empiece por `#` y termine por `.` (notar que el segundo `.` está *escapado* por la \; la ER `.*` implica 0 o más caracteres cualquiera)

Repetición Podemos repetir una regexp usando `\{ \}`

Constructor	Propósito
`\{`n`\}`	concuerda con exactamente n ocurrencias de la RE previa
`\{`n`,\}`	concuerda con al menos n ocurrencias de la RE previa
`\{`n`, `m`\}`	concuerda con entre n y m ocurrencias de la RE previa

Ejemplos:

- `a\{5\}`: 5 ocurrencias del carácter `a`
- `.\{5,\}`: al menos 5 ocurrencias de cualquier carácter

Expresiones regulares extendidas

Los sistemas UNIX actuales admiten extensiones a las expresiones regulares básicas:

- debemos usar `egrep`, `grep -E`, `sed -r`

ER	concuerda con
`+`	una o más ocurrencias de la RE anterior
`?`	cero o una ocurrencia de la RE anterior

Además, `\( \)` y `\{ \}` se reemplazan por `( )` y `{ }`

- Ejemplos:
  - `ab+c` concuerda con `abc`, `abbc`, pero no con `ac`
  - `ab?c` concuerda con `ac`, `abc`, pero no con `abbc`
- Para usar los caracteres `(`, `)`, `{` o `}` *escaparlos* con `\`

Alternancia El carácter `|` permite alternar entre 2 o más RE

- `(a|b)c` concuerda con `ac` o `bc`

Etiquetado Las RE que se ponen entre `( )` quedan etiquetadas, y podemos hacer referencia a esos elementos mediante $\backslash n$, con n el número de la etiqueta

- Ejemplos:
  - (.)oo\1 concuerda con `moom`, `noon`, pero no con `moon`
  - (.)oo\1-(.)aa\1\2 concuerda con `moom-paamp`

Otros caracteres Además de los ya vistos, pueden usarse otros metacaracteres:

ER	concuerda con
\n, \r, \t	LF, CR y tab (no siempre funcionan)
[:space:]	caracteres en blanco ([\t\n\r\f\v])
[:blank:]	espacio y tabulado
[:alnum:] o \w	caracteres alfánuméricos (letras y números)
[:digit:]	dígitos
[:alpha:]	alfabéticos
[:upper:]	mayúsculas
[:lower:]	minúsculas
[:xdigit:]	dígitos hexadecimales
[:punct:]	signos de puntuación
[:cntrl:]	caracteres de control
[:graph:]	caracteres imprimibles (sin espacio)
[:print:]	caracteres imprimibles (con espacio)
\<, \>	inicio/fin de palabra
\b	posición entre palabras
\B	posición en medio de una palabra

- `[[:upper:]]bc` concuerda con `Abc`, pero no `abc`
- `\babc\b` concuerda con `ab abc df`, pero no con `abcdef`
- `\Babc\B` concuerda con `ababcdf`, pero no con `ab abc df`

Más ejemplos

1. `\w+@\w+\.\w+((\.\w+)*)?` concuerda con direcciones de e-mail
2. `(0[1-9]|[12][0-9]|3[01])-(0[1-9]|1[012])-(19|20)[0-9]{2}` concuerda con fechas en el formato *dd-mm-yyyy* (años entre el 1900 y 2099)
3. `[-+]?([0-9]*\.)?[0-9]+([eE][-+]?[0-9]+)?` concuerda con números en punto flotante (con o sin exponente)

Ejemplos de uso con `sed`:

```
$ echo .ªbc1234def"| sed -r "s/[0-9]+/NUMERO/"
abcNUMERODdef
$ echo .ªbc1234def"| sed -r 's/[0-9]+/<&>/'
abc<1234>def
# En el siguiente ejemplo, notar que las ER intentan siempre
reconocer la secuencia más larga posible
$ echo "000x111x222x333"| sed 's/x.*x/<&>/'
000<x111x222x>333
```

```
# Eliminar blancos a principio y al final de línea y sustituir
más de un blanco seguido por uno solo
$ sed -r "s/^␣+// ; s/␣+$// ; s/␣␣+/␣/g"fich
# Pon los 4 primeros caracteres de cada línea al final
de la misma
$ sed -r 's/^(.{4,4})(.*)/\2\1/' fich
# Cambia de minúsculas a mayúsculas la primera letra de
cada palabra
$ sed -r 's/\<./\u&/g'
# Convierte DOS newlines (CR/LF) a formato Unix (LF)
$ sed 's/^M$//'[4]
# también funcionaría
$ sed 's/\r//'
```

Para más información: Regular-expressions.info

5.2. Comandos para el procesamiento de textos

Además de los ya vistos (`vi`, `grep`, `sed`) existen una serie de comandos para manejar ficheros de texto, como `tac`, `rev`, `nl`, `head`, `tail`, `sort`, `uniq`, `expand`, `fmt`, `cut`, `paste`, `tr`, `join`, `split`, `wc`, `od` o `awk`

- también se conocen como *filtros*: obtienen su entrada de la entrada estándar (o un fichero) y envían la salida a la salida estándar:

  ```
  sort < archivo.txt | head -3 > otro_archivo.txt
  ```

- casi todos estos comandos tienen, entre otras opciones, las siguientes dos:
  - `--help` muestra una pequeña ayuda y sal
  - `--version` muestra la versión del comando y sal
- también podemos saber más del comando a través de la página de manual o de `info`

Comandos simples

Existe una serie de comandos simples para realizar operaciones concretas sobre ficheros de texto

[4] Para introducir un carácter de control, como `^M`, tenemos que pulsar primero `Ctrl-V` y luego el carácter, en este caso `Enter`

- Ordena las líneas alfabéticamente: `sort`
- Escribe partes seleccionadas de un fichero a la salida estándar: `cut`
- Une texto de varios ficheros: `paste`
- Formatea párrafos: `fmt`
- Borra y/o reemplaza caracteres: `tr`
- Elimina líneas repetidas: `uniq`
- Combina varios ficheros: `join`
- Divide un fichero en ficheros más pequeños: `split`
- Muestra el principio/final de un fichero: `head`/`tail`
- Muestra el fichero al revés: `tac`, `rev`
- Muestra el número de líneas, palabras y bytes de un fichero: `wc`
- Añade números de línea: `nl`
- Convierte `TABs` en espacios: `expand`
- Muestra un fichero en diferentes formatos: `od`

Comentaremos brevemente cada uno de ellos

sort ordena alfabéticamente líneas de texto y las muestra en la salida estándar
Formato:

`sort` [opciones] *fichero*

Algunas opciones:

- `-b` ignora blancos al principio de línea
- `-f` no distingue mayúsculas/minúsculas
- `-r` orden inverso
- `-m` mezcla ficheros previamente ordenados
- `-n` ordena numéricamente

- `-k` *POS1[, POS2]* ordena según los campos desde *POS1* a *POS2*, o el final si no está *POS2* (el primer campo es 1)

Ejemplos:

```
$ cat nombres.txt
María Pérez
luis Andión
Adriana Gómez
jorge pena
$ sort nombres.txt
Adriana Gómez
María Pérez
jorge pena
luis Andión

$ sort -f nombres.txt
Adriana Gómez
jorge pena
luis Andión
María Pérez
$ sort -f +1 +0 nombres.txt #Obsoleto (no usar)
luis Andión
Adriana Gómez
jorge pena
María Pérez
$ sort -f -k 2,2 nombres.txt
luis Andión
Adriana Gómez
jorge pena
María Pérez
```

cut Escribe partes seleccionadas de un fichero a la salida estándar; puede usarse para seleccionar columnas o campos de un fichero específico
Formato:

`cut` [opciones] *fichero*

Algunas opciones:

- `-b`, `-c`, `-f` corta por bytes, caracteres o campos, respectivamente
- `-d` fija el carácter delimitador entre campos (por defecto, `TAB`)

Ejemplos:

```
$ cat nombres-ord.txt
Luis Andión
Adriana Gómez
Jorge Pena
María Pérez

$ cut -c 1-7 nombres-ord.txt
Luis An
Adriana
Jorge P
María P
$ cut -c 1-5,9-10 nombres-ord.txt
Luis ió
AdriaGó
Jorgena
Maríare
$ cut -d ' ' -f 1 nombres-ord.txt
Luis
Adriana
Jorge
María
```

paste Permite unir texto de varios ficheros, uniendo las líneas de cada uno de los ficheros

Formato:

`paste` [opciones] *fichero1 [fichero2]* ...

Algunas opciones:

- `-s` pega los ficheros secuencialmente, en vez de intercalarlos
- `-d` especifica los carácteres delimitadores en la salida (por defecto, `TAB`)

Ejemplos:

```
$ cat nombres.txt
Luis
Adriana
Jorge
María
```

```
$ cat apellidos.txt
Andión
Gómez
Pena
Pérez
$ paste nombres.txt apellidos.txt
Luis     Andión
Adriana  Gómez
Jorge    Pena
María    Pérez
$ paste -d ' ' nombres.txt apellidos.txt
Luis Andión
Adriana Gómez
Jorge Pena
María Pérez
$ paste -s -d '\t\n' nombres.txt
Luis   Adriana
Jorge  María
```

fmt Formatea cada párrafo, uniendo o separando líneas para que todas tengan el mismo tamaño
Algunas opciones:

- `-`*n* o `-w` *n* pone la anchura de las líneas a *n* (por defecto, 75)
- `-c` conserva la indentación a principio de línea y alinea a la izquierda la segunda línea
- `-s` las líneas pueden dividirse, no unirse
- `-u` uniformiza el espaciado entre palabras

Ejemplo:

```
$ cat quijote.txt
En un lugar de la Mancha, de    cuyo nombre no
quiero acordarme, no ha mucho tiempo
que vivía un
hidalgo    de los de lanza en astillero, adarga
antigua, rocín flaco y galgo corredor.

$ fmt -w 45 -u quijote.txt
En un lugar de la Mancha, de cuyo nombre
```

```
no quiero acordarme, no ha mucho tiempo
que vivía un hidalgo de los de lanza en
astillero, adarga antigua, rocín flaco y
galgo corredor.
```

tr Borra caracteres o reemplaza unos por otros
Formato:

`tr` [opciones] *set1 set2*

Algunas opciones:

- `-d` borra los caracteres especificados en *set1*
- `-s` reemplaza caracteres repetidos por un único carácter

Ejemplos:

```
$ tr 'a-z' 'A-Z' < quijote.txt
EN UN LUGAR DE LA MANCHA, DE CUYO NOMBRE...
$ tr -d ' ' < quijote.txt
EnunlugardelaMancha,decuyonombre...
$ tr au pk < quijote.txt
En kn lkgpr de lp Mpnchp, de ckyo nombre...
$ tr lcu o < quijote.txt | tr -s o
En on ogar de oa Manoha, de oyo nombre
```

uniq Descarta todas (menos una) las líneas idénticas sucesivas en el fichero
Formato:

`uniq` [opciones] *fichero*

Algunas opciones:

- `-d` muestra las líneas duplicadas (sin borrar)
- `-u` muestra sólo las líneas sin duplicación
- `-i` ignora mayúsculas/minúsculas al comparar
- `-c` muestra el número de ocurrencias de cada línea
- `-s` n no compara los n primeros caracteres
- `-f` n no compara los n primeros campos

- `-t` *c* usa el carácter *c* como separador de campos (por defecto, espacio o tabulado)

Ejemplo:

```
$ cat nombres.txt
Julio Lorenzo
Pedro Andión
Celia Fernández
Celia Fernández
Juan Fernández
Enrique Pena
$ uniq nombres.txt
Julio Lorenzo
Pedro Andión
Celia Fernández
Juan Fernández
Enrique Pena
$ uniq -f 1 -c nombres.txt
  1 Julio Lorenzo
  1 Pedro Andión
  3 Celia Fernández
  1 Enrique Pena
```

join Permite combinar dos ficheros usando campos: busca en los ficheros por entradas comunes en el campo y une las líneas; los ficheros deben estar ordenados por el campo de unión

Formato:

`join` [opciones] *fichero1 fichero2*

Algunas opciones:

- `-i` ignora mayúsculas/minúsculas
- `-1` *FIELD* une en el campo *FIELD* (entero positivo) de *fichero1*
- `-2` *FIELD* une en el campo *FIELD* de *fichero2*
- `-j` *FIELD* equivalente a `-1` *FIELD* `-2` *FIELD*
- `-t` *CHAR* usa el carácter *CHAR* como separador de campos
- `-o` *FMT* formatea la salida (*M.N* fichero *M* campo *N*, 0 campo de unión)

- `-v` N en vez de la salida normal, muestra las líneas que no se unen del fichero N
- `-a` N además la salida normal, muestra las líneas que no se unen del fichero N

Ejemplo:

```
$ cat nombres1.txt
Luis Andión
Adriana Gómez
Jorge Pena
María Pérez
$ cat nombres2.txt
Pedro Andión
Celia Fernández
Julio Lorenzo
Enrique Pena
$ join -j 2 nombres1.txt nombres2.txt
Andión Luis Pedro
Pena Jorge Enrique
$ join -j 2 -o 1.1 2.1 0 nombres1.txt nombres2.txt
Luis Pedro Andión
Jorge Enrique Pena
```

split Divide un fichero en ficheros más pequeños; los ficheros más pequeños se nombran a partir del *prefijo* especificado (*prefijo*aa, *prefijo*ab,...)
Formato:

`split` [opciones] *fichero prefijo*

Si no se pone *fichero*, o se pone - se lee la entrada estándar
Algunas opciones:

- `-l` n pone n lineas en cada fichero de salida (por defecto 1000)
- `-b` n pone n bytes en cada fichero de salida
- `-C` n pone en cada fichero de salida tantas líneas completas como sea posible sin sobrepasar n bytes
- `-d` usa números en vez de letras para el nombre de los ficheros de salida

Ejemplo:

```
$ split -l 2 quijote.txt quij
$ ls quij*
quijaa quijab quijac quijote.txt
$ cat quijaa
En un lugar de la Mancha, de cuyo nombre
no quiero acordarme, no ha mucho tiempo
$ cat quijac
galgo corredor.
$ split -l 2 -d quijote.txt quij
$ ls quij*
quij00 quij01 quij02 ...
```

head Muestra el principio de un fichero
Formato:

`head` [opciones] *fichero*

Algunas opciones:

- `-n` N ó `-`N muestra las primeras N líneas
- `-c` N muestra los primeros n bytes
- `-v` le añade una línea de cabecera, con el nombre del fichero

Ejemplo:

```
$ head -n 2 -v quijote.txt
==>quijote.txt <==
En un lugar de la Mancha, de cuyo nombre
no quiero acordarme, no ha mucho tiempo
```

tail Muestra el final de un fichero
Algunas opciones:

- `-n` N ó `-`N muestra las últimas N líneas (por defecto, 10)
- `+`N muestra de la línea N al final
- `-c` N muestra los últimos N bytes
- `-f` hace que `tail` corra en un lazo, añadiendo líneas a medida que el fichero crece (útil para cuando queremos ver como se modifica un fichero)

- `--retry` útil con `-f`; aunque el fichero no exista o sea inaccesible continua intentando hasta que puede abrirlo
- `-v` le añade una línea de cabecera, con el nombre del fichero

Ejemplo:

```
$ tail -n 2 -v quijote.txt
==>quijote.txt <==
astillero, adarga antigua, rocín flaco y
galgo corredor.
```

tac, rev `tac` imprime el fichero de la última a la primera línea (opuesto a `cat`); `rev` invierte las lineas del fichero
Ejemplos:

```
$ tac quijote.txt
galgo corredor.
astillero, adarga antigua, rocín flaco y
que vivía un hidalgo de los de lanza en
no quiero acordarme, no ha mucho tiempo
En un lugar de la Mancha, de cuyo nombre

$ rev quijote.txt
erbmon oyuc ed ,ahcnaM al ed ragul nu nE
opmeit ohcum ah on ,emradroca oreiuq on
ne aznal ed sol ed ogladih nu aíviv euq
y ocalf nícor ,augitna agrada ,orellitsa
.roderroc oglag
```

wc Muestra el número de líneas, palabras y bytes de un fichero
Formato:

`wc` [opciones] *fichero*

Algunas opciones:

- `-l` muestra sólo el número de lineas
- `-w` muestra sólo el número de palabras
- `-c` muestra sólo el número de bytes
- `-L` muestra la longitud de la línea más larga

Ejemplo:

```
$ wc quijote.txt
  5 33 178 quijote.txt
$ wc -l quijote.txt
5 quijote.txt
$ wc -w quijote.txt
33 quijote.txt
$ wc -c quijote.txt
178 quijote.txt
```

nl Añade números de línea; `nl` considera los ficheros separados en *páginas lógicas*, cada una de ellas con una cabecera, cuerpo y pie, cada una de estas secciones se numera de forma independiente, y la numeración se reinicia para cada página; los comienzos de cabecera, cuerpo y pie de cada página se marcan, respectivamente, con `\:\:\:`, `\:\:` y `\:`
Formato:

`nl` [opciones] *fichero*

Algunas opciones:

- `-b`, `-h` o `-f` *ESTILO* indica el estilo de numeración para cuerpo, cabecera o pie, que puede ser:
  - `a`: numera todas las líneas
  - `t`: numerar sólo las líneas no vacías (por defecto para el cuerpo)
  - `p` *REGEXP*: numera sólo las líneas que concuerdan con *REGEXP*
  - `n`: no numera ninguna línea (por defecto para cabecera y pie)
- `-v` n inicia la numeración en n (por defecto, 1)
- `-i` n incrementa los números por n (por defecto, 1)
- `-p` no reinicia la numeración al principio de cada página
- `-s` *STRING* una STRING para separar los números de línea del texto (por defecto ' ')

Ejemplo:

```
$ nl -s 'q ' quijote.txt
  1q En un lugar de la Mancha, de cuyo nombre
  2q no quiero acordarme, no ha mucho tiempo
  3q que vivía un hidalgo de los de lanza en
  4q astillero, adarga antigua, rocín flaco y
  5q galgo corredor.
```

expand Convierte `TABs` en espacios; útil debido a que la representación del `TAB` puede ser diferente en distintos sistemas
Formato:

`expand` [opciones] *fichero* ...

Algunas opciones:

- `-t` *n* reemplaza cada `TAB` por *n* espacios (por defecto, 8)
- `-i` solo reemplaza los `TABs` de principio de línea

Ejemplos:

```
$ cat hola.c
main() {
        for(i=0; i<10;i++)
                printf("Hola mundo!\n");
}
$ expand -t 2 hola.c
main() {
  for(i=0; i<10;i++)
    printf("Hola mundo!\n");
}
```

El comando `unexpand` hace la operación contraria

od Muestra un fichero en octal, hexadecimal o otros formatos; en cada línea muestra (en la primera columna) el *offset*
Formato:

`od` [opciones] *fichero*

Algunas opciones:

- `-t` *TIPO* especifica el formato de la salida (por defecto octal): `o` para octal, `x` para hexadecimal, `d` para decimal, `c` para caracteres ASCII, `a` para caracteres con *nombre...*
- `-A` *TIPO* especifica el formato del offset (por defecto octal): `o`, `x`, `d` como antes, `n` para que no aparezca
- `-w` *BYTES* número de bytes por linea (por defecto 16)

Ejemplo:

```
$ od -t x -A x quijote.txt
000000 75206e45 756c206e 20726167 6c206564
000010 614d2061 6168636e 6564202c 79756320
000020 6f6e206f 6572626d 206f6e0a 65697571
...
```

awk

Lenguaje diseñado para procesar datos basados en texto; el nombre AWK deriva de los apellidos de los autores: Alfred V. Aho, Peter J. Weinberger, y Brian W. Kernighan

- los administradores de sistemas utilizan `awk` para procesar los ficheros de configuración y logs de los sistemas
- estos ficheros, normalmente, se organizan en forma de *tabla* (líneas compuestas por campos)
  - `awk` es ideal para tratar esos ficheros
- sólo veremos algunos de los aspectos más importantes del uso de `awk` para el manejo de ficheros de texto

Funcionamiento básico `awk` lee el fichero que se le pase como entrada (o la entrada estándar) línea a línea, y sobre cada línea ejecuta una serie de operaciones

Ejemplo:

```
# echo -e interpreta "\nçomo un retorno de carro,
# lo que envía 2 líneas al comando awk
$ echo -e "\n"| awk '{ print "Hola mundo!"}'
Hola mundo!
Hola mundo!
```

Formas de ejecutar `awk` Podemos usar `awk` de varias formas:

- En la línea de comandos:

  `awk` *`PROGRAMA fichero_entrada`*

- Escribiendo el programa en un fichero:

  `awk -f` *`FICHERO_PROGRAMA fichero_entrada`*

- Ejecutando el *`FICHERO_PROGRAMA`* como un script:

  poner

  ```
  #!/usr/bin/awk -f
  ```

  al principio de *`FICHERO_PROGRAMA`*

Ejemplos:

```
$ echo '{ print "Hola mundo!"}' > hola.awk
$ echo -e "\n"| awk -f hola.awk
Hola mundo!
Hola mundo!
$ echo '#!/usr/bin/awk -f' > hola.awk
$ echo '{ print "Hola mundo!"}'» hola.awk
$ chmod +x hola.awk
$ echo -e "\n"| ./hola.awk
Hola mundo!
Hola mundo!
```

Estructura de un programa `awk` Un programa `awk` tiene tres secciones:

1. Parte inicial, que se ejecuta sólo una vez, antes de empezar a procesar la entrada:

   `BEGIN {` *`operaciones`* `}`

2. Parte central, con instrucciones que se ejecutan para cada una de las líneas de la entrada; tienen en siguiente formato:

   /`PATRÓN`/ `{` *`operaciones`* `}`

   las *`operaciones`* se realizan sólo sobre las líneas que verifiquen la REGEXP indicada en *`PATRÓN`*

- si ponemos `!/`*PATRÓN*`/` las operaciones se ejecutan en las líneas que no concuerden con el patrón

3. Parte final, se efectúa sólo una vez, después de procesar la entrada:

   `END { `*operaciones*` }`

Manejo de ficheros de texto `awk` divide las líneas de la entrada en campos:

- la separación entre campos la determina la variable `FS` (por defecto, uno a más blancos o `TAB`s
- las variables $1, $2, . . . , $N contienen los valores de los distintos campos
  - $0 contiene la línea completa

Ejemplos:

```
$ ls -ldh * |\
> awk '{print "Fichero ", $8, .°cupa ", $5, "bytes"}'
Fichero proba ocupa 36 bytes
Fichero uy_hist1_nodos.txt ocupa 9,1K bytes
Fichero vimbook-OPL.pdf ocupa 3,7M bytes


$ df -h | sort -rnk 5,5 |\
> awk 'BEGIN { print "Nivel de ocupación"}\
> /^\/dev\/hd/ {print "Partición ",$6,": ",$5}\
> END { print "Terminado"}'
Nivel de ocupación
Partición /home : 87% ocupación
Partición /mnt/hda2 : 51% ocupación
Partición / : 38% ocupación
Terminado

$ # Usando un fichero
$ cat ocupacion.awk
BEGIN {
     print "Nivel de ocupación"
}
/^\/dev\/hd/ {
     print "Partición ",$6,": ", $5
```

```
}
END { print "Terminado" }
$ df -h | sort -rnk 5,5 | awk -f ocupacion.awk
```

Variables predefinidas: `awk` tiene un conjunto de variables predefinidas, como `FS` que nos permite especificar el separador de campos

Esas variables son:

Nombre	Significado
`FS`	Carácter separador entre campos de entrada (por defecto, blanco o tabulado)
`NR`	Número de registros de entrada
`NF`	Número de campos en el registro de entrada
`RS`	Carácter separador entre registros de entrada (por defecto, nueva línea)
`OFS`	Carácter separador entre campos en la salida (por defecto, un espacio en blanco)
`ORS`	Carácter separador entre registros de salida (por defecto, nueva línea)
`FILENAME`	Nombre del fichero abierto

Ejemplo:

```
$ cat usuarios.awk
BEGIN {
   FS = ":"; OFS = "-->"; ORS = "\n============\n";
}
{
   print NR, $1, $5
}
$ awk -f usuarios.awk /etc/passwd
...
37 -->tomas -->Tomás Fernández Pena,,,
============
38 -->caba -->José Carlos Cabaleiro Domínguez,,,
============
...
```

Otras características `awk` es un lenguaje completo:

- permite definir variables de usuario

- permite realizar operaciones aritméticas sobre las variables
- permite utilizar condiciones, lazos, etc.
- permite definir funciones

La sintaxis de `awk` es prácticamente idéntica a la del lenguaje C

- podemos usar `printf` en lugar de `print` (con la sintaxis de C)
- también podemos usar arrays

Ejemplos:

1. Lista el tamaño de los ficheros y el tamaño total

```
$ cat lista-ficheros.awk
BEGIN { total = 0; }
{
   total += $5;
   printf("Fichero %s ocupa %d bytes\n", $8,$5);
}
END {
   printf("Ocupación total = %d bytes\n", total);
}
$ ls -ld * | awk -f lista-ficheros.awk
Fichero ancestros.awk ocupa 370 bytes
Fichero hola.c ocupa 66 bytes
Fichero lista-ficheros.awk ocupa 143 bytes
Ocupación total = 579 bytes
```

2. Muestra una advertencia si el nivel de ocupación de una partición supera un límite

```
$ cat ocupacion2.awk
BEGIN { limite = 85; }
/^\/dev\/hd/ {
   if($5 >limite)
     printf("PELIGRO: el nivel de ocupación de %s es %s\n %",
$6, $5);
}
$ df -ah | tr -d '%' | awk -f ocupacion2.awk
PELIGRO: el nivel de ocupación de /home es 87 %
```

Paso de parámetros: es posible pasar parámetros en la llamada a `awk`
Ejemplo: Indicando el `PID` de un proceso obtiene el `PID` de todos sus ancestros (padres, abuelos, ...)

```
$ cat ancestros.awk
BEGIN { ind=0; }
function padre(p) {
   for(i=0; i <ind; i++)
     if(pid[i] == p) return(ppid[i]);
}
!/PID/ { pid[ind]=$3; ppid[ind]=$4; ind++; }
END {
   do {
     printf("%d --> ", proc); proc = padre(proc);
   } while(proc >= 1);
   printf("\n\n");
}
$ ps axl | awk -f ancestros.awk proc=4258
4258 --> 3326 --> 1 -->
```

Arrays asociativos: `awk` permite el uso de arrays asociativos, es decir, que pueden tener como índice una cadena de caracteres
Ejemplo

```
$ cat usuarios2.awk
BEGIN { FS = ":"}
{ nombre[$1] = $5; }
END {
  for(;;){
    printf("Nombre de usuario: ");
    getline user < ";
    if( user ==  )
      break;
    printf("<%s>: %s\n", user, nombre[user]);
  }
}
$ awk -f usuarios2.awk /etc/passwd
Nombre de usuario: tomas
<tomas>: Tomás Fernández Pena,,,
Nombre de usuario:
```

Funciones predefinidas En `awk` existen una serie de funciones predefinidas

- `getline`: lee la siguiente línea de la entrada, pudiendo asignarla a una variable
  - `getline` *variable* `<` *fichero*
    lee una línea de fichero y la mete en *variable*
  - `getline` *variable* `< "`
    lee una línea de la entrada estándar y la mete en *variable*
  - `"`*comando*`" | getline`
    coge la salida de comando y la pone en la variable $0, descomponiéndola en campos ($1, $2, ...)

  Ejemplo:

  ```
  $ awk 'BEGIN{ "date"| getline; print $4 }'
  15:16:59
  ```

- `system`: ejecuta un comando del sistema operativo; en caso de éxito retorna 0, y en caso de error retornará un valor distinto de cero
  Ejemplo:

  ```
  $ awk 'BEGIN {\
  > if (system("ls")!=0)\
  > printf (.Error de ejecución"); }'
  ```

6. Programación en Python

Además de la programación con bash, sed y awk, existen otros lenguajes adecuados para la creación de scripts de administración

Perl: lenguaje de propósito general originalmente desarrollado para la manipulación de textos

Python: alternativa a Perl, más limpio y elegante

Ruby: combina una sintaxis inspirada en Python y Perl con características de programación orientada a objetos

Los tres son lenguajes de propósito general

- Permiten programar aplicaciones de muy diversos tipos
- Veremos solo una introducción a sus principales características, centrándonos principalmente en Python

Un buen administrador de sistemas debería dominar al menos uno de ellos

6.1. Introducción a Python

Bash es complejo y el código Perl puede resultar demasiado "ofuscado"

- Python es una buena alternativa a los lenguajes de script tradicionales

Principales características

- Soporte de diversos paradigmas: imperativo, orientado a objetos y funcional
- Sistema de tipos dinámico y gestión automática de memoria
- Énfasis en la legibilidad
- Uso de identación para delimitar bloques de código
- Gran librería con módulos para múltiples tareas

Ejemplo sencillo:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Abre el fichero sólo lectura
try:
  f = open("/etc/passwd","r")
except IOError:
  print "No puedo abrir /etc/passwd"
else:
  # Lee las líneas en una lista
  lista = f.readlines()
  # Recorre e imprime la lista
  for l in lista:
    print l, # La coma elimina \n
  f.close()
```

6.2. Tipos de datos en Python

Además de los tipos "estándar" Python proporciona:

1. Listas: mutables, pueden contener tipos mezclados

   ```
   frutas=["naranjas", "uvas", 123, "limones", "uvas"]
   frutas.append("peras")
   frutas.remove(123)
   ```

```
frutas.remove("uvas") # [naranjas,limones,uvas,peras]
frutas[2:2] = ["fresas", "pomelos"] # inserta en pos 2
print frutas       # naranjas,limones,fresas,pomelos,uvas,peras
print len(frutas)  # 6
print frutas[0:3]  # naranjas, limones, fresas
print frutas[-3]   # pomelos
print frutas[1:-3] # limones, fresas
frutas.pop()       # Elimina el último elemento
del frutas[2:4]    # Elimina los elementos 2 y 3
frutas.sort()      # Ordena
print frutas       # [limones,naranjas,uvas]
a=list("hola")     # a=["h","o","l","a"]
"o" in a           # True
```

Las listas pueden enlazarse

```
a = [[0,1],[2,3]]
print a[1][1]   # 3
a.append([4,5])
print a[2][0]   # 4
del a[1]
print a         # [0,1], [4,5]
```

`range`: función built-in que genera listas de valores en secuencia:

```
l = range(5)         # l = [0, 1, 2, 3, 4]
l = range(2, 5)      # l = [2, 3, 4]
l = range(2, 10, 3)  # l = [2, 5, 8]
l = range(5, -5, -2) # l = [5, 3, 1, -1, -3]
a = sum(range(1,4))  # a = 6
```

Las listas son objetos mutables (string, enteros, etc. no)

```
a = 1      # nuevo objeto entero (1) al que a referencia
b = a      # a y b referencias al mismo objeto entero (1)
a += 5     # se crea un nuevo objeto 6 (1+5)
print b    # 1, b sigue referenciando al objeto 1
a = [1, 2] # nuevo objeto lista
b = a      # a y b referencias al mismo objeto lista
a[0] += 5  # se modifica el objeto (mutable)
print b    # [6, 2] b es modificado
```

Copia de listas

```
a = [1, 2] # nuevo objeto lista
b = a[:]   # a y b referencias objetos diferentes
a[0] += 5  # se modifica el objeto (mutable)
print b    # [1, 2] b no se modificado
c=list(a)  # otra forma
```

2. Tuplas: listas inmutables

```
y=("enero","febrero","marzo","abril", "mayo", "junio",\
"julio","agosto","septiembre","octubre","noviembre",\
"diciembre")  # Paréntesis opcionales
print y[3]    # Abril
```

3. Conjuntos (Sets): sin elementos duplicados

```
cesta=["naranjas", "uvas", "limones", "uvas"]
frutas=set(cesta)
print frutas  # naranjas,uvas,limones
a = set("abracadabra")
b = set("alacazam")
print a      # "a", "r", "b", "c", "d"
print a-b    # "r", "b", "d"
print a | b  # "a", "c", "b", "d", "m", "l", "r", "z"
print a & b  # "a", "c"
print a ^ b  # "b", "d", "m", "l", "r", "z"
```

4. Diccionarios

```
edad_de = {"Eva":23, "Ana":19, "Oscar":41}
print edad_de["Ana"]  # Imprime 19
edad_de["Eva"] = 18   # Cambia un valor
edad_de["Juan"] = 26  # Añade un elemento
del edad_de["Oscar"]  # Borra un elemento
edad_de.keys()        # ["Eva", "Juan", "Ana"]
edad_de.values()      # [18, 26, 19]
for key,value in edad_de.items():
  print key,"->",value
dict([("a",1),("b",2),("c",3)])  # {"a":1, "c":3, "b":2}
dict(a=1, b=2, c=3) # {"a":1, "c":3, "b":2}
```

Compresión de listas

```
x = [1, 2, 3, 4, 5, 6, 7, 8]
xx = [n ** 2 for n in x if n > 4]  # xx=[25, 36, 49, 64]
```

```
l = [0, 1, 2, 3]
m = ["a", "b"]
n = [s*v for s in m
         for v in l
         if v > 0]    # n = ["a", "aa", "aaa", "b", "bb", "bbb"]

dict([(x, x**2) for x in (2, 4, 6)]) # {2:4, 4:16, 6:36}
```

6.3. Control de flujo

Lazos

```
frutas=["naranjas", "uvas"]
for f in frutas:
  print f, len(f)  # naranjas, 8; uvas, 4

for i in range(len(frutas)):
  print i, frutas[i] # 0, naranjas; 1, uvas

nf = raw_input("Añade otra fruta: ")
while nf:             # Si la entrada no está vacía
  frutas.append(nf)   # añádela a la lista
  nf = raw_input("Añade otra fruta: ")
```

Condicionales

```
x = int(raw_input("Introduce un entero: "))
if x < 0:
  x = 0
  print "Negativo cambiado a 0"
elif x == 0:
  print "Cero"
else:
  print "Positivo"
```

Funciones

```
def compra(fr, nf="manzanas"):
  fr.append(nf)

frutas=[]     # También frutas=list()
```

```
compra(frutas, "peras")
compra(frutas)
compra(nf="limones", fr=frutas)
print frutas   # peras, manzanas, limones
```

Funciones con argumentos arbitrarios

```
def fun(*args, **kwargs):
  for arg in args: print arg
  for kw in kwargs.keys(): print kw, ":", kwargs[kw]
fun("peras", 1, manzanas=2, limones=3)
```

Salida:

```
peras
1
limones : 3
manzanas : 2
```

6.4. Orientación a objetos

```
class fruteria(object):
   """Ejemplo simple de clase"""
   def __init__(self, f):
       self.stock = list()
       self.stock.append(f)
   def compra(self, f):
       self.stock.append(f)
   def vende(self, f):
       if f in self.stock:
           self.stock.remove(f)
       else:
           print f, "no disponible"

def main():
   mi_fruteria = fruteria("pera")
   mi_fruteria.compra("manzana")
   print mi_fruteria.stock       # ["pera", "manzana"]
   mi_fruteria.vende("pera")
   mi_fruteria.vende("platano")  # platano no disponible
   print mi_fruteria.stock       # ["manzana"]
   mi_fruteria.vende("pera")     # pera no disponible
   print mi_fruteria.__doc__     # Ejemplo simple de clase
```

```
if __name__ == "__main__":
   main()
```

Herencia múltiple

Se permite herencia múltiple:

```
class fruteria(object):
    def que_vendo(self):
        print "Vendo frutas"

class carniceria(object):
    def que_vendo(self):
        print "Vendo carne"

# Herencia múltiple
class tienda(carniceria, fruteria):
    pass

# La clase carniceria está más a la
# izquierda en la deficición de tienda
tienda().que_vendo() # Vendo carne
```

Métodos y atributos privados

Los métodos o atributos privados se definen con dos guiones bajos antes del nombre (y no pueden terminar en dos guiones bajos)

```
class Ejemplo(object):
    def publico(self):
        print "Uno"
        self.__privado()

    def __privado(self):
        print "Dos"

ej = Ejemplo()
ej.publico()   # Imprime Uno Dos
ej.__privado() # Da un error
```

6.5. Procesamiento de textos

Muchos métodos de interes para manejar cadenas de texto

```
# Elimina caracteres y separa por espacios
l = "Hola que tal!".strip("!").split() # l=["Hola", "que", "tal"]
# Une utilizando un caracter
s = ",".join(l)  # s="Hola,que,tal"
#Cuenta el número de ocurrencias de un caracter
c = s.count(",") # c=2
# Reemplaza un caracter por otro
ss = s.replace(",", "\t")  # ss="Hola    que    tal"
# Separa por otro tipo de caracter, e invierte la lista
l=ss.split("\t")
l.reverse()     # l=["tal", "que", "Hola"]
# Localiza una subcadena en el string
c=ss.find("tal")  # c=9
c=ss.find("tall") # c=-1 (no encuentra la subcadena)
# Separa por líneas
ml = """Esto es
un texto con
varias lineas"""
l = ml.splitlines()  # l=["Esto es", "un texto con", "varias lineas"]
```

Expresiones regulares

```
import sys, re  # Módulo para REGEXPR
# Comprueba direcciones de e-mail
s=raw_input("Introduce un e-mail: ")
if re.match("\w+@\w+\.\w+((\.\w+)*)?", s):
  print "Dirección correcta"

# Busca URLs en un fichero de texto
try:
  f = open("fich.txt","r")
except IOError:
  print "No puedo abrir"
  sys.exit(1)
for l in f:
  # Busca todas las URLs en la línea actual
  # y guárdalas (sin http) en la lista h
  h = re.findall("http://([^\s]+)", l)
  if h:         # Si la lista no está vacía
    for w in h: # recorrela e imprime las URLs
        print w

# Separa un string en una lista
```

```
s = "Uno:Dos.Tres-Cuatro"
l = re.split("[:.-]", s)
```

6.6. Otros aspectos

- Funciones anónimas (lambda)

```
g = lambda x: x**2
print g(8)         # 64

def suma (n): return lambda x: x + n
f=suma(2)
g=suma(8)
print f(10), g(10) # 12, 18
print suma(5)(11)  # 16
```

- Métodos map, filter y reduce

```
foo = [2, 18, 9, 22, 17, 24, 8, 12, 27]
print filter(lambda x: x % 3 == 0, foo)
# [18, 9, 24, 12, 27]
print map(lambda x: x * 2 + 10, foo)
# [14, 46, 28, 54, 44, 58, 26, 34, 64]
print reduce(lambda x, y: x + y, foo)
# 139
```

- Iteradores

```
# Iterador implícito en el for
for i in "papanatas":
  print i,        # p a p a n a t a s

# Iterador explícito
it = iter("papanatas")
it.next()  # p
it.next()  # a
it.next()  # p
it.next()  # a
it.next()  # n
it.next()  # a
it.next()  # t
it.next()  # a
```

```
it.next()  # s
it.next()  # Error
```

- Generadores

```
a = xrange(1000000)           # a no es una lista
b = (n for n in a if n%2==0)  # b no es una lista
print b   # <generator object <genexpr> at 0xb77c939c>
for i in b: print i,  # 2 4 6 8 10 ...

def generador():
    i = 0
    while True:     # un iterador infinito
        yield i     # devuelve i en este punto
        i = i + 1
mi_gen = generador() # creamos el generador
mi_gen.next()        # 0
mi_gen.next()        # 1
mi_gen.next()        # 2
```

- Métodos especiales:

```
class miclase:
  def __init__(self, n1, n2):
    self.n1 = n1
    self.n2 = n2
  # Representación del objeto como string
  def __str__(self):
    return "Soy un miclase con: n1="
            +str(self.n1)+", n2="+str(self.n2)
  # Permite asignar nuevos atributos
  def __setattr__(self, name, val):
    self.__dict__[name] = val
  # Se llama con atributos no conocidos
  def __getattr__(self, name):
    return "No se lo que es "+name
o = miclase(2, 5)
print o    # Soy un miclase con: n1=2, n2=5
o.n3 = 5
print o.n3 # Imprime "5"
print o.n4 # Imprime "No se lo que es n4"
```

6.7. Subprocesos

El módulo `subprocess` permite lanzar subprocesos, por ejemplo, comandos del SO

```
import subprocess
# Ejecuta el comando df -h (sintaxis de línea de comandos)
subprocess.call("df -h", shell=True)
# Ejecuta ls /usr/ppp, redireccionando la salida estándar
# y de error. El código de salida a ret
ret=subprocess.call(["ls", "/usr/ppp"],
                    stdout=open("/dev/null","w"),
                    stderr=subprocess.STDOUT)
# Ejecuta df -h; la salida estándar va al objeto p
p=subprocess.Popen(["df", "-h"], stdout=subprocess.PIPE)
# Lee e imprime las líneas de la salida de df -h
out = p.stdout.readlines()
for line in out:
   print line,
```

6.8. Otros módulos de interés

os Uso de funcionalidades dependientes del SO

- `os.getlogin()` nombre de login del usuario
- `os.getloadavg()` carga media del sistema
- `os.getcwd()` obtiene el directorio actual
- `os.chdir(`*`path`*`)` cambia el directorio actual a *path*
- `os.listdir(`*`path`*`)` lista de todas las entradas del directorio *path*

os.path Manipulación de ficheros y/o directorios

- `os.path.isfile(`*`path`*`)` True si *path* es un fichero regular
- `os.path.split(`*`path`*`)` Divide *path* en directorio+fichero
- `os.path.splitext(`*`path`*`)` Divide *path* en nombre_fichero+ extensión
- `os.path.getsize(`*`path`*`)` Devuelve el tamaño de *path*

glob Expansión de nombres de ficheros estilo UNIX (*globbing*)

- `glob.glob(`*`expr`*`)` Lista de ficheros indicados por *expr* (puede contener comodines)

shutil Operaciones de alto nivel con ficheros

- `shutil.copy(`*`src`*`, `*`dst`*`)` Copia el fichero *src* al fichero o directorio *dst*
- `shutil.move(`*`src`*`, `*`dst`*`)` Mueve recursivamente un fichero o directorio

tempfile Genera ficheros y directorios temporales

- `tempfile.NamedTemporaryFile()` Crea un fichero temporal con nombre

optparse Parsea las opciones en línea de comandos (reemplazado por `argparse`)

gzip, bz2, zipfile, tarfile Manejo de fichero comprimidos

sys Parametros y funciones dependientes del sistema

- `sys.argv` Lista de argumentos en línea de comandos (`sys.argv[0]` es el nombre del script)
- `sys.exit(`*`[code]`*`)` Termina el script con código de salida *code*

6.9. Ejemplos

1. En un directorio, renombra *.xml a *.html

```
import os.path, glob, shutil, optparse
def main():
    p = optparse.OptionParser(description="Renombra XML a HTML",
                              usage="%prog [directory]")
    options, args = p.parse_args()
    if len(args) == 1:
        # Chequea que sea un directorio
        if not os.path.isdir(args[0]):
            print args[0] + " no es un directorio"
            sys.exit(1)
        try:
            os.chdir(args[0]) # Cambia al directorio
            # Recorre los ficheros .xml
            for f in glob.glob("*.xml"):
                # Construye el nuevo nombre y renombra los ficheros
                new = os.path.splitext(f)[0] + ".html"
                shutil.move(f, new)
        except:
```

```
            print "Hubo un problema ejecutando el programa."
    else:
        p.print_help()
if __name__ == "__main__":
    main()
```

2. Muestra información sobre un proceso en ejecución

```
from subprocess import Popen, PIPE
proc = raw_input("Proceso a chequear: ")
try:
    # Ejecuta el comando ps y obten la salida
    output = Popen("ps -edf | grep "+proc,shell=True,stdout=PIPE)
    procs = output.stdout.readlines()
    for procinfo in procs:
        # Separa la salida en campos
        info = procinfo.split()
        # Muestra los resultados
        print "\n\
        Ejecutable:\t", info[-1], "\n\
        Propietario:\t", info[0], "\n\
        PID:\t\t", info[1], "\n\
        PPID:\t\t", info[2], "\n\
        Hora inicio:\t", info[4], "\n"
except:
    print "Hubo un problema ejecutando el programa."
```

3. Realiza acciones sobre un tar, seleccionándolas de un menú

```
import tarfile, sys
try:
  f = True
  while f:
    # Abre el fichero tar (especificado como argumento)
    tar = tarfile.open(sys.argv[1], "r")

    # Presenta el menú y obtiene la selección
    selection = raw_input("""
    Selecciona
       1 para extraer un fichero
       2 para mostrar información sobre un fichero en """
         + sys.argv[1] + """
```

```
        3 para listar los ficheros de """ + sys.argv[1] +
        """
        4 para terminar""" + "\n")

    # Realiza la acción en función de la selección
    if selection == "1":
      filename = raw_input("Indica el fichero a extraer:  ")
      tar.extract(filename)
    elif selection == "2":
      filename = raw_input("Indica el fichero a inspeccionar:  ")
      for tarinfo in tar:
        if tarinfo.name == filename:
          print "\n\
          Nombre:\t", tarinfo.name, "\n\
          Tamaño:\t", tarinfo.size, "bytes\n"
    elif selection == "3":
      print tar.list(verbose=True)
    elif selection == "4":
      f = False
    else:
        print "Selección incorrecta"
except:
    print "Hubo un problema ejecutando el programa."
```

Referencias

- Python Official Website: página principal de Python
- Python Documentation: documentación diversa, tutoriales, etc.
- The Python tutorial: un buen sitio para empezar
- The Python Standard Library: la librería estándar
- Módulos útiles
- Índice alfabético de módulos
- Python para todos: tutorial en castellano

7. Introducción a Perl y Ruby

7.1. Perl

Principales aplicaciones de Perl:

- Administración de sistemas
- Desarrollo web
- Programación en red
- Desarrollo de GUI
- ...

Algunas características

- Combina características de shell, awk y sed con otros lenguajes de alto nivel
- Soporte de distintos paradigmas de programación (imperativa, orientada a objetos y funcional)
- Potente sistema de procesamiento de texto mediante expresiones regulares
- Enorme colección de módulos disponibles

Ejecución de un script Perl

- Directamente en la línea de comandos:

```
# Renombra *.txt a *-2010.txt
$ perl -e 'foreach (<*.txt>)
>  { s/\.txt$//; rename("$_.txt", "$_-2010.txt") }'
```

- En un script

```
#!/usr/bin/perl
use strict;   # Exige predeclarar las variables (my)
use warnings; # Avisa de posibles errores
#
# Abre el fichero de contraseñas y lee cada línea.
my $filename = "/etc/passwd";  # Nombre del fichero
open(FILE, "<", $filename)     # Abre el fichero (solo lectura)
```

```
  or die "No puedo abrir: $!"; # Termina si falla
while(my $line = <FILE>) {     # Lee cada línea
        print $line;
}
close(FILE);                  # Cierra el fichero
```

Tipos de datos en Perl

1. Escalares (números o strings)

```
$a = "manzanas";
$b = "peras";
print $a." y ".$b."\n"; # Muestra "peras y manzanas"
print "$a y $b\n";      # Muestra "peras y manzanas"
```

2. Arrays

```
@frutas = ("naranjas", "limones", "uvas");
print $frutas[2];       # uvas
($n, $l) = @frutas;     # $n="naranjas", $l="limones"
push(@frutas, "cocos"); # $frutas[3] = "cocos"
$c = pop(@frutas);      # $c = "cocos"
$nf = scalar(@frutas);  # $nf = 3
$fr = "@frutas";        # $fr = "naranjas limones uvas"
@fo = split(/ /, $fr);  # @fo = ("naranjas", "limones", "uvas")
```

3. Mapas (arrays asociativos)

```
%edad_de = {
    Eva  => 23,
    Ana  => 19,
    Oscar => 41
}
print $edad_de{Ana};    # Imprime 19
$edad_de{Eva}=18;       # Cambia un valor
$edad_de{Juan} = 26;    # Añade un elemento al mapa
```

4. Variables especiales

   - `$_` Variable por defecto (la mayoría de las funciones de Perl toman `$_` como argumento por defecto)
   - `@ARGV` array con los argumentos de la línea de comandos
   - `%ENV` Mapa con las variables de entorno

Control de flujo

Lazos

```
foreach (@frutas) { # Recorre el array
    print $_."\n";  # Imprime un elemento por
                    # línea. El punto concatena
}                   # dos strings.

print "\nAñade más frutas ";  # Imprime un mensaje
$a = <STDIN>;                # Lee de la entrada estándar
chop $a;                     # y elimina el \n
while ( $a ) {               # Si la entrada no está vacía
    push(@frutas, $a);       # añádela al array
    $a = <STDIN>; chop $a;   # y lee una nueva entrada
}
```

Condicionales

```
if ( not $tengo_manzanas ) {
  compra(\@frutas,"manzanas" ); # El array se pasa por
}                               # referencia
```

Alternativa:

```
unless ($tengo_manzanas) {
  compra(\@frutas,"manzanas");
}
```

También es válido:

```
compra(\@frutas,"manzanas") if not $tengo_manzanas;
```

Subrutinas

- Los parámetros se recogen en `@_`

```
sub compra {
    ( $array, $string ) = @_; # Los parametros se recogen
                              # como escalares
    push(@$array, $string);   # La referencia se convierte
                              # a array
}
```

Expresiones regulares

```
             # Sin argumentos, lee la entrada estandar
while(<>) {  # con argumentos, usa estos como nombres
             # de ficheros y los lee línea a línea
  print if /http:\/\//; # Muestra las líneas con http://
  print if s/ttx/txt/ig; # Muestra las líneas con "ttx"
                         # y hace el cambio por "txt"
                         # g=global, i=case insensitive
}

$string = "oCme mas futra";
$string =~ s/oCme/Come/;  # =~ Aplica sustitución a $string
$string =~ s/futr/frut/;
print $string; # Imprime "Come mas fruta"
```

Ejemplos

1. Muestra las terminaciones de los ficheros del directorio actual

   ```
   #!/usr/bin/perl
   use strict;
   use warnings;
   foreach (glob("*")) {       # Recorre los ficheros
       my @file = split(/\./); # Los separa por .
       my $term = pop(@file);  # Extrae el último elemento
       print "$term\n";
   }
   ```

2. En un directorio, renombra *.xml a *.html

   ```
   #!/usr/bin/perl
   use strict;
   use warnings;
   unless (scalar(@ARGV) == 1) {
     print "Necesito un directorio como argumento\n"; exit 1;
   }
   if( not -d $ARGV[0] ) {
     print "$ARGV[0] no es un directorio\n"; exit 1;
   }
   # Cambia al directorio
   chdir $ARGV[0];
   ```

```
# Recorre los ficheros .xml
foreach my $file (glob "*.xml") {
  # Construye el nuevo nombre
  my $new = substr($file, 0, -3) . "html";
  # Renombra los ficheros
  rename $file, $new;
}
```

3. Lee un fichero de texto numerando las líneas no vacías

```
#!/usr/bin/perl
use strict;
use warnings;
open(my $fichero, "<", "f.txt")
     or die "No puedo abrir f.txt:$!";
my $nl="001"; # Entero de tres dígitos
while(<$fichero>) {
  if(!/^$/) {        # Sólo las líneas no vacías
    print "$nl $_";  # Pon un número de línea
    $nl++;
  }
  else {
    print "$_";      # Línea vacía sin número
  }
}
```

4. Script para añadir usuarios al sistema

```
use strict; use warnings;
# Módulo para leer parámetros de entrada
use Getopt::Long;
my $addusr = "/usr/sbin/adduser";
my $nombre=""; my $apellido="";
# Obtiene los parámetros
GetOptions("nombre=s" => \$nombre,
     "apellido=s" => \$apellido ) or uso();
# Comprueba los parámetros sean correctos
if( not $nombre or not $apellido ) {
    uso();
}
if ( $nombre !~ /^[a-zA-Z]+$/) {
    uso("El nombre debe ser alfabético");
```

```
}
if ( $apellido !~ /^[a-zA-Z]+$/) {
    uso("El apellido debe ser alfabético");
}

# Construye el username
my $username = lc( substr($apellido, 0, 1) . $nombre);
# Directorio HOME
my $home     = "/home/$username";
# Comando a ejecutar
my $comando = qq($addusr --home $home --disabled-password \\
            --gecos "$nombre $apellido" $username);
system $comando; # Ejecuta el comando

# Error e información de uso
sub uso {
  my ($msg) = @_;      # Recogo los parámetros
  if ($msg) {          # Si se pasa un mensaje de error,
     print "$msg\n\n"; # lo muestra
  }
  print "Usar: $0 --nombre Nombre --apellido Apellido\n";
  exit;
}
```

Referencias

- The Perl Directory: página principal de Perl
- Perl programming documentation: extensa documentación
- Comprehensive Perl Archive Network: módulos y documentación de Perl
- The CPAN search site: para buscar en el CPAN

7.2. Ruby

Lenguaje dinámico, de propósito general, creado a mediados de los 90 por Yukihiro "Matz"Matsumoto

- Expresiones regulares nativas similares a las de Perl

- Soporte de múltiples paradigmas: imperativo, orientado a objetos y funcional
- "Todo" es un objeto
- Amplia librería estándar

Ejemplo sencillo:

```
#!/usr/bin/ruby
=begin
Abre y lee un fichero
Se usa un bloque (entre do - end)
El identado no es necesario
El fichero se cierra
automáticamente al acabar el bloque.
=end
File.open("/etc/passwd", "r") do |f1|
  while linea = f1.gets
    puts linea
  end
end    # Fin del bloque
```

Tipos de datos en Ruby

1. Arrays

   ```
   frutas=[ "naranjas", "uvas", 123, "limones", "uvas" ]
   frutas<<"peras"     # Añade un string
   frutas.delete(123)
   frutas.uniq!        # Elimina elementos duplicados
   frutas.insert(2, %w{fresas pomelos}) # Inserta otro array
                                        # %w -> array de strings
                                        # sin usar comillas
   puts frutas # naranjas,uvas,fresas,pomelos,limones,peras
   puts frutas.length  # 5
   puts frutas[2][1]   # pomelos
   frutas.delete_at(2)
   frutas.insert(3, "cerezas", "kiwis")  # Inserta
   frutas.sort! # Ordena ``in-place''
   puts frutas  # cerezas, kiwis, limones, naranjas, peras, uvas
   ```

2. Rangos

```
nums = -1..9
puts nums.include?(10) # false (10 no en el rango)
puts nums === 0        # true (0 en el rango)
puts nums.first        # -1
puts nums.last         # 9
puts nums.to_a         # [-1,0,1,2,3,4,5,6,7,8,9]
puts nums.to_s         # "-1..9"
array = nums.reject {|i| i < 7}
puts array             # [7, 8, 9]
```

3. Arrays asociativos

```
edad_de = {'Eva'=>23, 'Ana'=>19, 'Oscar'=>41}
puts edad_de['Ana']     # Imprime 19
edad_de['Eva'] = 18     # Cambia un valor
edad_de['Juan'] = 26    # Añade un elemento
edad_de.delete('Oscar') # Borra un elemento
```

Control de flujo

Lazos

```
frutas=["naranjas", "uvas"]

# Bloque usando do-end
frutas.each do |f|
  puts "#{f}:#{f.length}"  # naranjas:8
end                           # uvas:4

print "Añade otra fruta: "
nf = gets.chomp    # Lee stdin y elimina el \n
while nf != ""     # Si la entrada no está vacía
  frutas<<nf.to_s  # añádela a la lista
  print "Añade otra fruta: "
  nf = gets.chomp
end

# Bloque usando llaves
3.times { |i| puts i }   # 0, 1, 2
```

Condicionales

```
print "Introduce un entero: "
x = gets.chomp.to_i
if x < 0
  x = 0
  puts "Negativo cambiado a 0"
elsif x == 0
  puts "Cero"
else
  puts "Positivo"
end

# Forma unless
unless x == 0
  puts x
end
# Case
scale = 8
case scale
  when    0: puts "lowest"
  when 1..3: puts "medium-low"
  when 4..5: puts "medium"
  when 6..7: puts "medium-high"
  when 8..9: puts "high"
  when   10: puts "highest"
  else       puts "off scale"
end
```

Funciones

```
# Argumento con valor por defecto
def compra(fr, nf="manzanas")
  fr<<nf
end
# Número de argumentos variable
def compram(fr, *nf)
  # Recorro todos los argumentos
  nf.each { |f| fr<<f }
end

frutas=[]
```

```
# Los paréntesis no son obligatorios
compra frutas, "peras"
# Usa el valor por defecto
compra(frutas)
# Usa múltiples argumentos
compram(frutas, "limones", "naranjas")
puts frutas   # peras, manzanas, limones, naranjas
```

Expresiones regulares

```
# Comprueba direcciones de e-mail
print "Introduce un e-mail: "
s = gets.chomp
if /\w+@\w+\.\w+((\.\w+)*)?/.match(s)
  puts "Dirección correcta"
end

# Busca URLs en un fichero de texto
# Abre el fichero de solo lectura
# comprobando excepciones
begin
  f = File.open("fich.txt","r")
rescue Exception => msg
  print "No puedo abrir --> ", msg, "\n"
  exit(1)
end

# Expresión regular a buscar (\s == [:space:])
urlreg = /http:\/\/([^\s]+)/
nl=1
f.each do |l|
  # Busca todas las URLs en la línea actual
  # e imprimelas
  l.scan(urlreg) { |m| print "Línea #{nl}-><#{m}>\n" }
  nl+=1
end
f.close

# Corrige un string
s = "oCme más futra"
s.gsub!("oCme", "Come")
```

```
s.gsub!("futr", "frut")
puts s  # Imprime "Come más fruta"
# Separa un string en una lista
s = "Uno:Dos.Tres-Cuatro"
l=s.split(/[:.-]/)
```

Ejemplos

1. En un directorio, renombra *.xml a *.html

   ```
   # Módulo con utilidades para ficheros
   require 'fileutils'
   # Comprueba argumentos
   if ARGV.length < 1
     puts "Necesito un directorio como argumento"
     exit
   end
   dir=ARGV[0]

   # Chequea que sea un directorio
   unless File.directory?(dir)
     puts dir+" no es un directorio"
     exit
   end

   # Recorre los ficheros .xml
   begin
     # Cambia al directorio
     FileUtils.cd(dir)
     Dir.glob("*.xml") do |f|
       # Construye el nuevo nombre
       new = File.basename(f, ".xml")+".html"
       # Renombra los ficheros
       File.rename(f, new)
     end
   rescue Exception => msg
     puts "Error: "+msg
   end
   ```

2. Muestra información sobre un proceso en ejecución

```
print "Proceso a chequear: "
proc = gets.chomp
begin
  # Ejecuta el comando ps y obten la salida
  output = `ps -edf|grep #{proc}`
  # Separa la salida en campos
  procinfo = output.split()

  # Muestra los resultados
  puts "Ejecutable  : #{procinfo[7]}"
  puts "Propietario : #{procinfo[0]}"
  puts "PID         : #{procinfo[1]}"
  puts "PPID        : #{procinfo[2]}"
  puts "Hora inicio : #{procinfo[4]}"
rescue Exception => msg
  puts "Error: "+msg
end
```

3. Busca recursivamente ficheros que cumplen un patrón

```
# Módulo adicional
require 'find'
print "Directorio inicial: "
searchpath = gets.chomp
print "Patrón de búsqueda: "
pattern = gets.chomp
# Busca recursivamente
Find.find(searchpath) do |path|
  # Comprueba si el patrón corresponde con el fichero
  if File.fnmatch(pattern, File.basename(path))
    # Muestra el nombre del fichero
    puts  "Fichero             : " +  File.basename(path)
    # Información sobre el fichero
    stat = File.stat(path)
    # Muestra los permisos en octal
    printf("Permisos            : %o\n", stat.mode)

    # Muestra el UID y el GID del propietario
    print "UID del propietario : "
    puts stat.uid
    print "GID del propietario : "
    puts stat.gid
```

```
    # Muestra el tamaño del fichero
    print "Tamaño (bytes)      : "
    puts stat.size
    puts "---------------------------------------"
  end
end
```

Referencias

- Página principal de Ruby
- Ayuda y documentación para Ruby
- Core API docs para Ruby 1.8.7
- Ruby en 20 minutos

Tema 3: Actividades administrativas básicas

Administración de Sistemas e Redes

Tomás Fernández Pena

tf.pena@usc.es

Índice

1. Comandos básicos para la gestión de procesos

En el tema anterior vimos como ejecutar comandos del shell:

- otros comandos ajenos al shell se ejecutan igual

En cada momento se están ejecutando un gran número de procesos:

- procesos de sistema (kernel, *daemons*)
- procesos de usuarios

En esta sección trataremos la gestión de los procesos que se están ejecutando:

- listar procesos en ejecución
- detener y matar procesos
- controlar la prioridad de ejecución

1.1. Ver los procesos en ejecución

Existen varias herramientas para ver los procesos en ejecución, la más importante es el comando `ps`

`ps` (***process status***)

lista los procesos con su PID, datos de usuario, tiempo, identificador del proceso y linea de comandos usada

```
$ ps
  PID TTY          TIME CMD
 6368 pts/0    00:00:00 bash
 7441 pts/0    00:00:00 ps
```

sin opciones, `ps` sólo muestra los procesos lanzados desde el terminal actual y con el mismo `EUID` que el usuario que lo lanzó

Opciones de ps `ps` tiene un gran número de opciones, que se pueden especificar de 3 maneras:

1. opciones UNIX: pueden agruparse y se preceden por un guión: `ps -ef`
2. opciones BSD: pueden agruparse y van sin guión: `ps uxa`
3. opciones largas GNU: precedidas de dos guiones: `ps --user tomas`

Algunas opciones:

- `-e` o `ax`: muestra todos los procesos
- `-u` (o `U` o `--user`) *usuario*: muestra los procesos de un usuario
- `u`: salida en formato usuario
- `j`: salida en formato *job* (muestra PID, PPID, etc.)
- `-f` o `l`: salida en formato largo
- `f`: muestra un árbol con la jerarquía de procesos
- `k` (o `--sort`) *campo*: ordena la salida por algún campo (p.e. `ps uxak rss`)
- `-o` (o `o` o `--format`) *formato*: permite definir el formato de salida `ps -o ruser,pid,comm=Comando`

para más opciones ver la página de manual de `ps`

Ejemplo:

```
$ ps axu
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   1516   536 ?        S    09:43   0:00 init [2]
root         2  0.0  0.0      0     0 ?        S    09:43   0:00 [migration/0]
root         3  0.0  0.0      0     0 ?        SN   09:43   0:00 [ksoftirqd/0]
root         4  0.0  0.0      0     0 ?        S    09:43   0:00 [migration/1]
.............
tomas     6475  0.1  4.9 140180 50920 ?        Sl   09:51   0:18 /usr/lib/mozilla-thunde
tomas     6528  0.2  3.6 116396 37948 ?        Sl   10:01   0:25 /usr/lib/mozilla-firef
```

en este ejemplo:

- `%CPU` y `%MEM`: porcentajes de uso de CPU y memoria
- `VSZ`: memoria virtual del proceso, en KBytes
- `RSS`: tamaño de la memoria residente (*resident set size*) en KBytes
- `STAT`: estado del proceso; puede ser:

Código	significado
D	*Uninterruptible sleep* (usualmente IO)
R	Ejecutándose(*running*) o en cola de ejecución
S	*Interruptible sleep* (p.e. esperando un evento)
T	Detenido
Z	Proceso *zombie*

cuando se usa formato BSD puede aparecer otro código acompañando al principal:

Código	significado
<	alta prioridad
N	baja prioridad
L	páginas bloqueadas (*locked*) en memoria
s	líder de sesión
l	multi-threaded
+	proceso en foreground

pstree muestra el árbol de procesos (similar a `ps f`)

```
init-+-acpid
     |-atd
     |-bonobo-activati
     |-clock-applet
```

```
|-cron
|-cupsd
|-dbus-daemon-1
|-dcopserver
|-dirmngr
|-2*[esd]
|-events/0-+-aio/0
|          |-ata/0
|          |-ata/1
|          |-kblockd/0
|          |-khelper
|          `-pdflush
|-events/1-+-aio/1
|          |-kacpid
|          |-kblockd/1
|          `-pdflush
|-exim4
|-famd
|-firefox-bin---wvMime---ggv
...
```

top

ps da una versión estática de los procesos

- top nos da una lista actualizada a intervalos

```
top - 17:34:08 up  7:50,  6 users,  load average: 0.12, 0.31, 0.27
Tasks: 111 total,   1 running, 110 sleeping,   0 stopped,   0 zombie
Cpu(s):  6.2% us,  2.0% sy,  0.0% ni, 91.0% id,  0.0% wa,  0.8% hi,  0.0% si
Mem:   1026564k total,   656504k used,   370060k free,    65748k buffers
Swap:  2048248k total,        0k used,  2048248k free,   336608k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 6130 root      15   0 63692  48m 9704 S  8.7  4.9   8:03.34 XFree86
 6341 tomas     15   0 14692 8852 6968 S  4.3  0.9   1:55.13 metacity
 6349 tomas     16   0 32792  14m 9232 S  1.3  1.5   0:41.60 gnome-terminal
 6019 tomas     15   0  7084 3184 1896 D  0.3  0.3   0:23.22 famd
 6401 tomas     15   0 16756 8280 6856 S  0.3  0.8   0:02.49 geyes_applet2
 6427 tomas     15   0 18288  10m 8112 S  0.3  1.0   0:09.04 wnck-applet
 7115 tomas     15   0 26312  13m  11m S  0.3  1.4   0:00.61 kio_uiserver
 7390 tomas     15   0 45016  30m  18m S  0.3  3.0   0:38.69 kile
    1 root      16   0  1516  536  472 S  0.0  0.1   0:00.61 init
    2 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    ......................
```

- en la cabecera nos muestra un resumen del estado del sistema
  - hora actual, tiempo que el sistema lleva encendido, el número de usuarios conectados y la carga media del sistema para los últimos 1, 5, y 15 minutos
  - número total de tareas y resumen por estado
  - estado de ocupación de la CPU y la memoria
- por defecto, los procesos se muestran ordenados por porcentaje de uso de CPU (los más costosos arriba)
- pulsando `h` mientras se ejecuta `top`, obtenemos una lista de comandos interactivos
- para salir, `q`
- Algunos campos de top
  - VIRT: Tamaño total del proceso (código, datos y librerías compartidas cargadas), VIRT=SWAP+RES
  - SWAP: Memoria que ha sido *swapped out* o que aún no ha sido cargada
  - RES: Memoria residente (RAM ocupada por el proceso)
  - CODE y DATA: Memoria ocupada por el código y datos (datos y pila, pero no librerías compartidas) del proceso
  - SHR: Memoria compartida (memoria que puede ser compartida con otros procesos)
  - P: Última CPU usada (SMP)
  - nFLT: Número de fallos de página para el proceso

strace

Muestra las llamadas al sistema realizadas por un proceso en ejecución

- Ejemplo de un `strace` sobre un `top` en ejecución

```
$ strace top
gettimeofday({1195811866, 763977}, {4294967236, 0}) = 0
open("/proc/meminfo", O_RDONLY)         = 3
fstat64(3, {st_mode=S_IFREG|0444, st_size=0, ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7f5
```

```
read(3, "MemTotal:      2066348 kB\nMemFre"..., 1024) = 728
close(3)                                = 0
munmap(0xb7f55000, 4096)                = 0
open("/proc", O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY) = 3
fstat64(3, {st_mode=S_IFDIR|0555, st_size=0, ...}) = 0
fcntl64(3, F_SETFD, FD_CLOEXEC)         = 0
getdents(3, /* 52 entries */, 1024)     = 1020
getdents(3, /* 64 entries */, 1024)     = 1024
stat64("/proc/1", {st_mode=S_IFDIR|0555, st_size=0, ...}) = 0
open("/proc/1/stat", O_RDONLY)          = 4
read(4, "1 (init) S 0 1 1 0 -1 4194560 44"..., 1023) = 185
close(4)
......
```

Ejecución en segundo plano

Por defecto, los comandos corren en primer plano (*foreground*): el shell espera a que termine el comando antes de aceptar uno nuevo

- para ejecutar un comando en segundo plano (*background*) hacerlo con &

  ```
  $ sleep 10
  $ sleep 10 &
  ```

- para terminar un proceso en foreground `Ctrl-C`
- para pausar un comando en foreground usar `Ctrl-Z`
  - `bg` pasa el proceso a background
  - `fg` lo devuelve a foreground
- Ejemplo:

  ```
  $ sleep 20
  Ctrl-Z
  [3]+ Stopped     sleep 20
  $ bg
  [3]+ sleep 20 &
  $ fg
  sleep 20
  ```

- El comando `jobs` permite ver la lista de comandos (*jobs*) en background lanzados desde el shell, así como su estado (`fg` y `bg` pueden referirse a uno de los jobs)

```
$ gedit nada.txt &; sleep 100 &
$ jobs
[2]- Running     gedit nada.txt &
[3]+ Running     sleep 100 &
$ fg 3
sleep 100
Ctrl-Z
[3]+ Stopped     sleep 100
$ jobs
[2]- Running     gedit nada.txt &
[3]+ Stopped     sleep 100
$ bg 3
[3]+ sleep 100 &
$ jobs
[2]- Running     gedit nada.txt &
[3]+ Running     sleep 100 &
```

1.2. Señalización de procesos

El comando básico para enviar señales a un proceso es `kill`

- Ctrl-C y Ctrl-Z son atajos para enviar señales SIGINT (2) y SIGTSTP (20)
- `kill -l` lista el conjunto de señales

```
$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     17) SIGCHLD
18) SIGCONT     19) SIGSTOP     20) SIGTSTP     21) SIGTTIN
22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO
30) SIGPWR      31) SIGSYS      ....
```

- para ver su significado, ver `man 7 signal`

Sintaxis de `kill`

`kill` *`[señal]`* `PID`

- *señal* puede indicarse mediante el número o el código:

    - `kill -9` y `kill -KILL` son equivalentes
- las señales más comunes son:
    - `SIGHUP` (1): cuelgue del terminal o muerte del proceso controlador
    - `SIGTERM` (15): mata el proceso permitiéndole terminar correctamente
    - `SIGKILL` (9): mata el proceso sin permitirle terminar
    - `SIGSTOP` (19): para el proceso
    - `SIGCONT` (18): continúa si parado
    - `SIGINT` (2): interrupción de teclado (Ctrl-C)
    - `SIGTSTP` (20): stop de teclado (Ctrl-Z)
    - `SIGQUIT` (3): salida de teclado (Ctrl-\)

Algunas características de las señales:

- La señal que se envía por defecto es `TERM` (15)
    - los procesos pueden ignorar esta señal y no terminar
    - las señales `KILL` (9) y `STOP` (19) no pueden ignorarse
- En bash, cuando enviamos una señal `SIGHUP` a un shell, este se lo reenvía a todos sus hijos
- Cuando cerramos un terminal en un entorno gráfico, o abandonamos una sesión, se envía un `SIGHUP` a todos sus hijos
- La mayoría de los demonios (*daemons*) responden a la señal `SIGHUP` volviendo a leer sus ficheros de configuración:
    - en vez de matar y reiniciar un demonio podemos hacer un `kill -HUP` para reiniciarlo

Ejemplos

```
$ yes >/dev/null &
[1] 9848
$ yes >/dev/null &
[2] 9849
$ ps
 PID TTY      TIME  CMD
9834 pts/7 00:00:00 bash
```

```
9848 pts/7 00:00:02 yes
9849 pts/7 00:00:01 yes
9850 pts/7 00:00:00 ps
$ kill -STOP 9849
[2]+ Stopped          yes >/dev/null
$ jobs
[1]- Running          yes >/dev/null &
[2]+ Stopped          yes >/dev/null
$ kill -CONT 9849
$ jobs
[1]- Running          yes >/dev/null &
[2]+ Running          yes >/dev/null &
$ kill -KILL 9848
$ kill -1 9849
[1]- Matado          yes >/dev/null
[2]+ Colgar          yes >/dev/null
```

Otros comandos

pgrep busca en la lista de procesos para localizar el PID a partir del nombre (similar a `ps | grep`)

- Ejemplo:

```
$ pgrep -u root sshd # PID del proceso sshd de root
```

pkill permite enviar señales a los procesos indicándolos por nombre en vez de por PID

- Ejemplo:

```
$ pkill -9 proceso
```

- si hay varios procesos con el mismo nombre los mata a todos
- en vez de un nombre admite un patrón (p.e. pkill ´l.*´)
  - tener cuidado con su uso (es fácil matar procesos de forma errónea)

killall similar a pkill, pero no admite patrones en el nombre, y tiene otras opciones

nohup normalmente, cuando salimos de un login shell (`logout`) o cerramos una un terminal, se envía una señal `SIGHUP` a todos los procesos hijos[1]:

- si lanzamos un proceso en background y salimos de la sesión el proceso se muere al morir el shell desde el que lo iniciamos

El comando `nohup` permite que un lanzar un comando ignorando las señales `SIGHUP`

`nohup` *`comando`*

- la salida del comando se redirige al fichero `nohup.out`

exec `exec` ejecuta un comando reemplazando al shell desde el que se lanza Ejemplos:

```
$ yes > /dev/null &
[1] 14724
$ yes > /dev/null &
[2] 14725
$ ps
  PID TTY      TIME  CMD
 7083 pts/3 00:00:00 bash
14724 pts/3 00:00:02 yes
14725 pts/3 00:00:02 yes
14726 pts/3 00:00:00 ps
$ pgrep yes
14724
14725
$ pkill -9 yes
$ ps
  PID TTY      TIME  CMD
 7083 pts/3 00:00:00 bash
14730 pts/3 00:00:00 ps
[1]- Matado         yes > /dev/null
[2]+ Matado         yes > /dev/null
```

Más ejemplos:

[1] en teoría, en bash, podemos fijar el comportamiento de la shell modificando la opción `huponexit` con `shopt`

```
$ nohup yes > /dev/null &
[1] 9620
$ kill -HUP 9620
$ ps
 PID TTY      TIME  CMD
8293 pts/5 00:00:00 bash
9620 pts/5 00:00:13 yes
9621 pts/5 00:00:00 ps
$ kill 9620
[1]+ Terminado         nohup yes > /dev/null
```

1.3. Manejo de la prioridad y recursos de un proceso

Cuando un proceso se ejecuta, lo hace con una cierta *prioridad*

- las prioridades van desde -20 (prioridad más alta) a 19 (prioridad más baja)
- por defecto, los procesos se ejecutan con prioridad 0
  - un usuario normal solo puede asignar prioridades más bajas (números positivos)
  - `root` puede asignar prioridades más altas (números negativos)
- los comandos para manejo de prioridades son `nice` y `renice`

`nice`

Permite lanzar un comando con una cierta prioridad

- Sintaxis

  `nice -n` *`ajuste comando`*

  la prioridad por defecto se modifica por *`ajuste`*

- Ejemplo:

```
$ nice -n 10 abiword &     # disminuye la prioridad en 10
$ ps -o pid,pri,ni,stat,cmd
  PID PRI  NI STAT CMD
 7133  24   0 Ss   bash
 7431  14  10 SN   abiword
 7552  23   0 R+   ps -o pid,pri,ni,stat,cmd
```

```
$ nice -n -1 abiword
nice: no se puede establecer la prioridad: Permiso denegado
```

`renice`

Permite cambiar la prioridad de un proceso que está en ejecución

- Sintaxis:

  `renice` *`pri`* `[-p` *`pid`*`] [-g` *`pgrp`*`] [-u` *`user`*`]`

- las opciones son:
  - `-p` *`pid`* cambia la prioridad para el proceso especificado
  - `-g` *`pgrp`* cambia la prioridad para los procesos ejecutados por los usuarios que pertenecen al grupo con ID=*`pgrp`*
  - `-u` *`user`* cambia la prioridad para los procesos del usuario especificado
- Ejemplo:

```
$ abiword &
[1] 7681
$ renice 10 -p 7681
7681: old priority 0, new priority 10
$ renice 3 -u tomas
503: old priority 0, new priority 3
```

Control de los recursos de un proceso

El comando interno de bash `ulimit` permite controlar los recursos de los que dispone un proceso arrancado por el shell

- Sintaxis

  `ulimit [`*`opciones`*`] [`*`limite`*`]`

- Algunas opciones:
  - `-a` muestra los límites actuales
  - `-f` máximo tamaño de los ficheros creados por el shell (opción por defecto)
  - `-n` máximo número de ficheros abiertos

- `-s` máximo tamaño de la pila
- `-t` máximo tiempo de cpu
- `-S`/`-H` usa los límites *soft* y *hard*
  - el usuario puede incrementar su límite *blando*, pero sin superar el límite duro
  - estos límites pueden ser fijados en el `/etc/profile`, `/etc/bash.bashrc`

- Para más información `help ulimit`
- Ejemplo: limitar el tamaño de los ficheros creados a 1 KByte

```
$ ulimit -f 1
```

1.4. Análisis básico del rendimiento del sistema

Además de `ps` y `top` existen comandos básicos que nos pueden mostrar el estado del sistema en cuanto a uso de CPU y consumo de memoria[2]

`uptime`

Muestra la hora actual, el tiempo que el sistema lleva encendido, el número de usuarios conectados y la carga media del sistema para los últimos 1, 5, y 15 minutos (lo mismo que en la primera línea de `top`)

- Ejemplo:

```
$ uptime
 20:25:03  up 25 days, 11:12, 13 users,  load average: 3.00, 3.07, 3.08
```

`w`

Además de la información dada por `uptime`, el comando `w` muestra información sobre los usuarios y sus procesos

- Ejemplo:

```
$ w
 20:24:52  up 25 days, 11:11, 13 users,  load average: 3.10, 3.09, 3.08
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
paula    pts/1    godello.dec.usc. 12:08pm  8:15m  8.39s  8.36s  ssh -p 1301 -X paula@p
```

[2] Veremos otros comandos de monitorización más complejos (como `vmstat` o `sar`) en la asignatura ASR II

```
paula    pts/2    godello.dec.usc. 12:09pm  7:30   0.11s  0.11s  bash
pichel   pts/4    -                11:08am  7.00s  0.33s  0.33s  -bin/tcsh
pichel   pts/5    -                 7:12pm 56:56   0.26s  0.16s  ssh usceljpc@sc.cesga.
pichel   pts/6    -                 4:35pm 21:15  16.61s  0.02s  /bin/sh reordena.sh ma
tomas    pts/8    jumilla.dec.usc.  8:24pm  0.00s  0.05s  0.02s  w
```

- Definiciones:
  - `LOGIN@` la hora a la que se conectó el usuario
  - `IDLE` tiempo que lleva ocioso el terminal
  - `JCPU` el tiempo de CPU consumido por los procesos que se ejecutan en el TTY
  - `PCPU` tiempo consumido por el proceso actual (el que aparece en la columna `WHAT`)

`free`

Muestra la cantidad de memoria libre y usada en el sistema, tanto para la memoria física como para el swap, así como los buffers usados por el kernel (similar a lo mostrado en la cabecera de `top`)

- Ejemplo:

```
$ free
             total       used       free     shared    buffers     cached
Mem:       3098556    2969388     129168          0     419300    1674492
-/+ buffers/cache:     875596    2222960
Swap:      6144736    3129376    3015360
```

- la columna `shared` no significa nada (obsoleta)
- Opciones:
  - `-b,-k,-m,-g` memoria en bytes/KBytes/MBytes/GBytes
  - `-t` muestra una línea con el total de memoria (física + swap)
  - `-s` *`delay`* muestra la memoria de forma continua, cada *`delay`* segundos

1.5. Herramientas gráficas

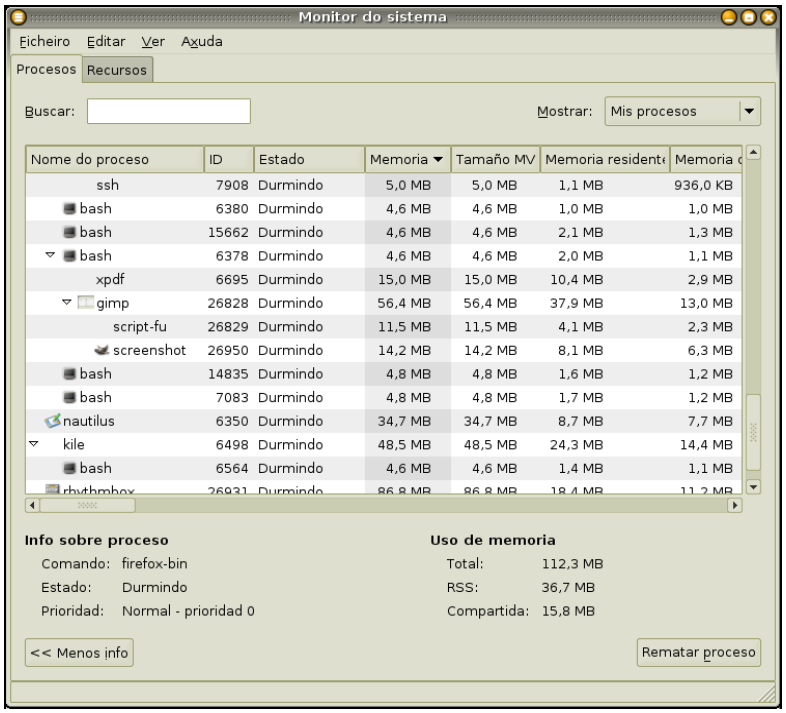
Además de los comandos comentados, si disponemos de un entorno X Windows podemos usar alguna herramienta gráfica en vez de `top` y `ps`:

- proporcionan una visión más clara y son fáciles de usar

Algunas herramientas son

- `gnome-system-monitor`: visor de procesos y monitorizador de recursos del sistema de GNOME
- KDE System Guard (`ksysguard`): gestor de tareas y monitor de rendimiento de KDE

Ejemplo: `gnome-system-monitor`



1.6. El directorio `/proc`

Pseudofilesystem que guarda información sobre el sistema y los procesos

- se inicializa durante el arranque
- está implementado en memoria y no se guarda en disco

- la estructura del directorio `/proc` depende de la versión del kernel
- los comandos vistos (`ps`, `top`, etc.) obtienen la información sobre los procesos de este directorio

Algunos ficheros y directorios son:

- `cpuinfo`: información estática de la CPU
- `meminfo`: información de uso de la memoria
- `partitions`: información sobre las particiones
- `filesystems`: sistemas de ficheros soportados por el kernel
- `version`: versión y fecha del kernel
- `bus/`: directorio con información de los buses PCI y USB
- `cmdline`: línea de arranque del kernel
- `devices`: dispositivos del sistema de caracteres o bloques
- `modules`: módulos del kernel
- `net/`: directorio con información de red
- `interrupts`: muestra el número de interrupciones por IRQ
- `ioports`: lista los puertos de entrada salida usados en el sistema

Además, existen un directorio por cada proceso, que se identifica con el PID del proceso, en el que se puede encontrar información sobre cada proceso, incluidos:

- el directorio desde que se invoco el proceso (enlace `cwd`)
- nombre del ejecutable (enlace `exe`) y la línea de comandos con la que fue invocado (fichero `cmdline`)
- entorno en que se ejecuta el proceso (fichero `environ`)
- estado del proceso (fichero `status`)
- descriptores de ficheros abiertos y archivos o procesos relacionados (directorio `fd`)
- mapa de memoria (fichero `maps`)

Nota: en el kernel 2.6 de Linux aparece un nuevo pseudofilesystem (`/sys`) que reemplaza al `/proc` en lo que se refiere a visualización y ajuste de dispositivos

2. Gestión del sistema de ficheros

UNIX tiene múltiples comandos para trabajar con ficheros y directorios: `ls`, `rm`, `cp`, `mv`, `mkdir`, `rmdir`, `touch`, `etc.`

- estos comandos tienen opciones que es importante conocer
- ver las páginas de manual para las distintas opciones

En esta sección trataremos:

- los diferentes tipos de ficheros y sus atributos
- los permisos de acceso para ficheros y directorios
- la creación de enlaces
- la localización de ficheros
- la creación de particiones y sistemas de ficheros

2.1. Tipos de ficheros y atributos

La mayoría de los sistemas de ficheros definen 7 tipos de ficheros:

Ficheros normales son los usuales; se crean con distintos programas (`vi`, `cp`, `touch`, `etc.`) y se borran con `rm`

Directorios contiene referencias a otros ficheros y directorios; se crean con `mkdir` y se borran con `rmdir` o `rm -r`

Ficheros de dispositivos de caracteres o bloques permiten la comunicación con el hardware y los periféricos; se crean con `mknod` y se borran con `rm`

- caracteres: entrada/salida byte a byte
- bloques: entrada salida en bloques de datos

Tuberías con nombre (*named pipes*) también llamados ficheros FIFO, permiten la comunicación entre procesos; se crean con `mknod` y se borran con `rm`

Sockets comunican procesos en la red; se crean con `socket()` y se borran con `rm` o `unlink()`

Enlaces simbólicos también llamados enlaces *blandos*: apuntador a otro fichero; se crean con `ln -s` y se borran con `rm`

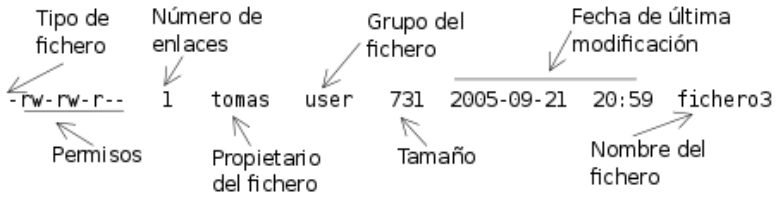
El comando `file` nos permite determinar el tipo de un fichero:

- para ficheros normales, distingue según contenido (fichero de imagen, pdf, ASCII, etc)
- Ejemplo:

```
$ file /dev/xconsole
/dev/xconsole: fifo (named pipe)
$ file fichero1
fichero1: PDF document, version 1.2
$ file fichero2
fichero2: Microsoft Office Document
$ file fichero3
fichero3: PNG image data, 750 x 686, 8-bit/color RGB, non-interlaced
```

Atributos de un fichero

Podemos ver los atributos de un fichero con `ls -l`



Indicador de tipo el primer carácter nos indica el tipo del fichero

Carácter	Tipo
-	fichero normal
d	directorio
l	enlace simbólico
c	fichero de dispositivo de caracteres
b	fichero de dispositivo de bloques
p	tubería
s	socket

Número de enlaces indica el número de nombres (enlaces duros) del fichero

- en el caso de un directorio, esto corresponde con el número de subdirectorios (incluidos `.` y `..`)

Tamaño es el tamaño en bytes

- con `ls -lh` se ve el tamaño de forma más legible
- el tamaño máximo de un fichero depende del filesystem usado

Fecha especifica la fecha de última modificación del fichero

- podemos actualizarla con el comando `touch`

Nombre la longitud máxima del nombre es de 255 caracteres

- evitar el uso de espacios y caracteres especiales como `*,$, ?, ´, ", /, \`

Permisos de ficheros y directorios

UNIX proporciona tres operaciones básicas para realizar sobre un fichero o directorio: lectura (`r`), escritura (`w`) y ejecución (`x`)

- Efecto sobre un fichero:
  1. lectura (`r`): permite abrir y leer el fichero
  2. escritura (`w`): permite modificar o truncar el fichero (para borrarlo, basta con que el directorio tenga permiso de escritura)
  3. ejecución (`x`): permite ejecutar el fichero (binario o script)
- Efecto sobre directorios:
  - ejecución (`x`): permite entrar en el directorio (pero no listar su contenido, ni crear ficheros o directorios)
  - lectura y ejecución (`rx`): permite listar el contenido del directorio (pero no crear ficheros o directorios)
  - escritura y ejecución (`wx`): permite crear, borrar o renombrar ficheros (pero no listar su contenido)
  - acceso total (`rwx`)

Los permisos se aplican en tres categorías:

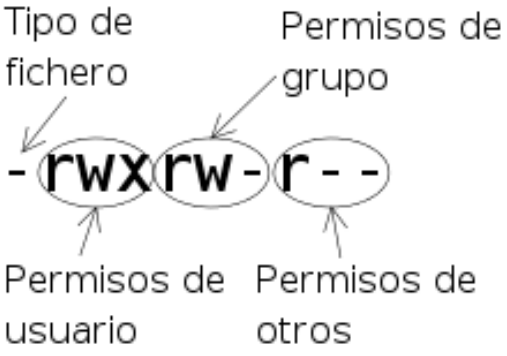
- Permisos de usuario (`u`): propietario del fichero (por defecto, el usuario que lo creó)

- Permisos de grupo (`g`): grupo del fichero (por defecto, grupo principal del usuario que lo creó)
- Permisos de otros (`o`): resto de usuarios

Cada usuario cae en uno solo de estas categorías:

- p.e. al propietario se le aplican los permisos de usuario, aunque sean más restrictivos que los de grupo

Los permisos se identifican con 9 caracteres:



Cambio de permisos

El comando para modificar los permisos es `chmod`

- Formato

  `chmod [-R]` *`operación ficheros`*

- `-R` indica acceso recursivo
- solo el propietario del fichero (o `root`) puede cambiar los permisos

`operación` indica como cambiar los permisos, y puede especificarse mediante símbolos o números:

- Permisos simbólicos: formato *`quien op permisos`*

  quien especificado por `u`, `g`, `o` o `a` para todos

  op puede ser `+` para añadir permisos, `-` para quitar o `=` para establecer

  permisos especificados por `r`, `w`, `x`

  Ejemplos:

- `chmod u+x temp.dat`
  añade permisos de ejecución para el usuario (manteniendo los permisos existentes)
- `chmod ug=rw,o=r temp.dat`
  lectura y escritura para usuario y grupo y sólo lectura para el resto
- `chmod -R =r *`
  pon, de forma recursiva, permisos sólo lectura para todos (ugo)
- `chmod a-x *.bak`
  quita el permiso de ejecución para todos
- `chmod g=u temp.dat`
  pon los permisos de grupo igual a los del usuario
- `chmod a= *`
  quita los permisos a todos

- Permisos numéricos:
  - *operación* se representa por un número **octal** de tres dígitos, para `u`, `g`, y `o` respectivamente
  - cada dígito vale:
    - 4 para `r`, 2 para `w` y 1 para `x`
    - para combinaciones, se suman:
      p.e. `rw` es 6, `rx` es 5 y `rwx` es 7

  Ejemplos:
  - `chmod 750 temp.dat`
    permisos `rwx` para usuario, `rx` para grupo y ninguno para otros
  - `chmod 043 temp.dat`
    ninguno para usuario, `r` para grupo y `wx` para otros

Permisos especiales

Además de `rwx` existen los permisos `setuid`/`setgid` (`s`) y `sticky bit` (`t`)

`setuid y setgid` están relacionados con los atributos de los procesos:

- cuando un proceso se crea se le asigna un UID/GID real y un UID/GID efectivo

UID/GID real identificadores de usuario y grupo del usuario que lanzó el proceso (y que puede matarlo)

UID/GID efectivos determinan las operaciones que el proceso puede hacer sobre los objetos del sistema

- por ejemplo, un proceso con UID efectivo 0 (`root`) puede manipular todos los ficheros del sistema

- lo normal es que los UID/GID normal y efectivo de un proceso coincidan

Podemos usar `ps` para ver los RUID/RGID y EUID/EGID

- `ps axo ruid,rgid,euid,egid,cmd` para ver números
- `ps axo ruser,euser,rgroup,egroup,cmd` para nombres

Los permisos `setuid`/`setgid` permiten que un proceso lanzado por un usuario se ejecute con EUID/EGID de otro usuario

- Ejemplo: el programa `passwd`

  ```
  -rwsr-xr-x 1 root root 25872 2005-07-25 23:15 /usr/bin/passwd
  ```

  cuando un usuario ejecuta `passwd` este proceso puede modificar el fichero `/etc/shadow` propiedad de root

Fijar `setuid`/`setgid`

- Forma simbólica

  `chmod u+s` fija `setuid`
  `chmod g+s` fija `setgid`

- Forma numérica: 4000 `setuid`, 2000 `setgid`

Ejemplo:

```
$ ls -l temp
-rw-r-----  1 tomas gac 0 2005-09-22 18:07 temp
$ chmod u+s temp; ls -l temp
-rwSr-----  1 tomas gac 0 2005-09-22 18:07 temp
$ chmod u+x temp; ls -l temp
-rwsr-----  1 tomas gac 0 2005-09-22 18:07 temp
$ chmod 6740 temp; ls -l temp
-rwsr-S---  1 tomas gac 0 2005-09-22 18:07 temp
```

IMPORTANTE: es peligroso poder ejecutar procesos con permisos de otro usuario

- debería evitarse el uso de ficheros `setuid`/`setgid`

sticky bit solo se usa en directorios

- permite crear ficheros en el directorio (si tiene permiso de escritura), pero solo los puede borrar:
  - el propietario del fichero
  - el propietario del directorio
  - el superusuario

Ejemplo:

```
$ ls -ld /tmp
drwxrwxrwt 15 root root 3072 2005-09-22 19:09 /tmp/
```

Para activar el `sticky bit`

- `chmod +t` *`dir`*
- `chmod 1`*`xxx dir`*

Permisos por defecto

Cuando se crea un fichero se cambian los permisos por defecto

- estos permisos pueden modificarse con `umask`

  `umask [`*`opciones`*`]` *`valor`*

  donde *`valor`* son tres dígitos que especifican los permisos para `u`, `g`, `o` según la tabla

Octal	Permisos	Octal	Permisos
0	`rwx`	4	`-wx`
1	`rw-`	5	`-w-`
2	`r-x`	6	`--x`
3	`r--`	7	`---`

- Opciones (dependen de la versión de shell):
  - `-S` muestra los permisos por defecto
- Ejemplo[3]

  ```
  $ umask 027
  $ umask -S
  u=rwx,g=rx,o=
  ```

[3] Comandos que crean ficheros como `touch` o `vi` no ponen permiso de ejecución aunque lo diga `umask`

Cambio de usuario/grupo

Los comandos `chown` y `chgrp` permiten cambiar el propietario y grupo de un fichero

- sólo `root` puede cambiar el propietario
- el grupo puede cambiarse a otro al que pertenezcamos

Formato:

```
chown [opciones] propietario ficheros
chgrp [opciones] grupo ficheros
chown [opciones] propietario:grupo ficheros
```

Algunas opciones

- `-R` recorre recursivamente los subdirectorios
- `-v` (*verbose*) indica las operaciones que realiza

2.2. Enlaces

Permiten referirse a un fichero con otro nombre

Dos tipos:

- Enlaces duros: asignan otro nombre al fichero
  - crean una entrada en el directorio apuntando al mismo nodo-i que el fichero original
  - el fichero no se borra hasta que se borran todos sus enlaces duros
  - no se puede enlazar con ficheros de otra partición
- Enlaces blandos: un fichero que apunta al original
  - si el fichero se borra, el enlace permanece sin apuntar a nada
  - no tienen problema con las particiones

Comando `ln`

Permite crear enlaces

- Formato
  `ln [-s] [`*`opciones`*`] `*`destino`*` [`*`enlace`*`]`
  `ln [-s] [`*`opciones`*`] `*`destino1 destino2 ... [directorio`*`]`
- con `-s` se crean enlaces blandos
- si no se pone nombre del enlace se usa el del *`destino`*
- Ejemplos:

```
pepe@jumilla:~$ ln /home/luis/fich1 fichhard
pepe@jumilla:~$ ln /home/luis/fich2 /home/luis/fich3 .
pepe@jumilla:~$ ls -il /home/luis/fich* ./fich*
293993 -rw-r--r-- 2 luis luis 12 Sep 22 20:19 ./fich2
294011 -rw-r--r-- 2 luis luis 12 Sep 22 21:18 ./fich3
293838 -rw-r--r-- 2 luis luis 13 Sep 22 20:17 ./fichhard
293838 -rw-r--r-- 2 luis luis 13 Sep 22 20:17 /home/luis/fich1
293993 -rw-r--r-- 2 luis luis 12 Sep 22 20:19 /home/luis/fich2
294011 -rw-r--r-- 2 luis luis 12 Sep 22 21:18 /home/luis/fich3
pepe@jumilla:~$ ln -s /home/luis/fich4 blando
pepe@jumilla:~$ ls -il /home/luis/fich4 blando
294012 -rw-r--r-- 1 luis luis 12 Sep 22 21:22 /home/luis/fich4
277445 lrwxrwxrwx 1 pepe pepe 17 Sep 22 21:22 blando -> /home/luis/fich4
```

2.3. Localización de ficheros

Una tarea común de administración es la búsqueda de ficheros que verifiquen ciertas propiedades

- buscar ficheros muy grandes
- buscar ficheros de un determinado usuario
- mostrar los ficheros que se hayan modificado en los últimos 2 días
- buscar ficheros `setuid`/`setgid`

El comando básico para hacer esto es `find`

Comando `find`

Busca a través de la jerarquía de directorios ficheros que cumplan determinado criterio

- Formato

  `find [`*`directorio_de_búsqueda`*`] [`*`expresión`*`]`

- Ejemplo:

  - Busca desde `/etc` los ficheros de tipo socket

    ```
    find /etc -type s
    ```

  - Busca desde `/etc` y `/usr/share` los ficheros que se llamen `magic` o `passwd`

    ```
    find /etc /usr/share -name magic -o -name passwd
    ```

  - Muestra desde el directorio actual todos los ficheros de forma recursiva

    ```
    find
    ```

La *`expresión`* tiene los siguientes componentes:

- opciones: modifican la forma de operación de `find`
- criterio de búsqueda
- acciones: especifica que hacer con los ficheros que encuentra
- operadores: permiten agrupar expresiones

Opciones de `find` normalmente se colocan al principio de la expresión

Opción	Efecto
`-maxdepth` *n*	desciende como máximo *n* directorios
`-mindepth` *n*	empieza a buscar a partir del nivel *n*
`-depth`	procesa el contenido del directorio antes que el propio directorio
`-daystart`	para medidas con tiempo, empieza desde el principio del día actual
`-mount` o `-xdev`	no pasa a otras particiones

Criterios de busqueda

Criterio	Efecto
`-name` *patrón*	busca ficheros que coincidan con el patrón (pueden usarse comodines, escapados)
`-wholename`	permite incluir nombres con el `path`
`-iname`	igual que name pero no distingue mayúsculas/minúsculas
`-regex`	igual pero usa REGEXPR
`-type` *tipo*	busca por tipo de fichero (`b, c, d, p, l, s, f`)
`-size [+/-]n[bck]`	busca por tamaño (tamaño igual, mayor o menor que *n* con `b`=bloques, `c`=bytes y `k`=KB)
`-perm [+/-]`*permisos*	busca por permisos (sin nada, permisos exactos, `-` todos los permisos y `+` alguno de los permisos)
`-user` *nombre*	busca por propietario
`-uid` *n*, `-gid` *n*	busca por UID/GID
`-nouser`, `-nogroup`	busca ficheros con prop./grupo no válidos

Busqueda por atributos temporales

Criterio	Efecto
`-atime [+/-]`*n*	busca ficheros cuya fecha de acceso para lectura coincide con, es anterior a (`+`) o es posterior a (`-`) *n* días
`-mtime [+/-]`*n*	lo mismo, pero con la fecha de última modificación del fichero
`-ctime [+/-]`*n*	lo mismo, pero con la fecha en que se cambió el estado del fichero
`-amin/-mmin/` `-cmin [+/-]`*n*	lo mismo, pero ahora *n* representa minutos
`-newer` *file*	busca ficheros modificados más recientemente que *file*
`-anewer` *file*	ficheros con último acceso más reciente que la modificación de *file*
`-cnewer` *file*	ficheros con cambio de estado más reciente que la modificación de *file*

Acciones de `find` `find` permite realizar distintas acciones con los ficheros que encuentra

- mostrar su nombre (acción por defecto)
- mostrar otra información del fichero
- ejecutar un comando sobre el fichero

Acción	Descripción
`-print`	imprime el nombre de los ficheros que encuentra (acción por defecto)
`-ls`	imprime el nombre de los ficheros con formato de listado largo
`-exec` *`comando`* `\{\} \;`	ejecuta *`comando`* sobre los ficheros encontrados
`-ok` *`comando`* `\{\} \;`	igual que `-exec` pero pregunta antes de ejecutar *`comando`*
`-prune`	si directorio no desciende por el (permite ignorar directorios)

los caracteres `{}` se refieren al fichero que `find` acaba de encontrar y `;` indica el fin del comando

Operadores de `find` permiten agrupar expresiones

Operador	Descripción
`expr1` `-a` *`expr2`*	AND (*`expr2`* no se evalúa si *`expr1`* es falsa)
`expr1` *`expr2`*	igual que `-a`
`expr1` `-o` *`expr2`*	OR (*`expr2`* no se evalúa si *`expr1`* es cierta)
`!` *`expr1`*	NOT (cierto si *`expr`* falsa)
`(` *`expr1`* `)`	agrupan expresiones (hay que escapar los paréntesis)

Ejemplos con `find`

- `find . -maxdepth 1 -user david`
  busca ficheros, sólo en el directorio actual, propiedad de `david`
- `find / -name \*.html -ls`
  busca, en todo el sistema de ficheros, ficheros terminados en `.html` y muestra un listado largo

- `find /home/httpd/html ! -name \*.html`
  busca, desde `/home/httpd/html`, los ficheros que no acaben en `.html`
- `find /home -size +2500k -mtime -7`
  busca, desde `/home`, ficheros más grandes de 2500KB que hayan sido modificados en los últimos 7 días
- `find /home -iname \*.bak -ok rm \{\} \;`
  busca ficheros terminados en `.bak` (sin distinguir mayúsculas/minúsculas) y pregunta si se quiere borrar
- `find /home -iname \*.bak -exec mv \{\} /BAK \;`
  busca ficheros terminados en `.bak` y muevelos a `/BAK`
- `find / -wholename ´/home´ -prune -o -name \*.bak -ls`
  busca excluyendo el directorio `/home`
- `find . -perm 022`
  encuentra ficheros con permisos `-----w--w-`
- `find . -perm +022`
  encuentra ficheros escribibles por `grupo` **O** `otros`
- `find . -perm -022`
  encuentra ficheros escribibles por `grupo` **Y** `otros`
- `find . -perm -g=w,o=w`
  idéntico al anterior
- `find /home/httpd/html -name \*.html \`
  `> -exec grep -l expired \{\} \;`
  lista los nombres de los `.html` que tengan la palabra `expired`

Este último ejemplo funciona, pero es muy ineficiente (¿por qué?)

- otra forma de hacer lo mismo

  ```
  grep -l expired $(find /home/httpd/html -name \*.html)
  ```

si el número de ficheros `.html` es muy grande `grep` puede tener problemas

- podemos usar `xargs`

  ```
  find /home/httpd/html -name \*.html |\
  > xargs grep -l expired
  ```

Otros comandos para localizar ficheros

Existen otros comandos para la localización de ficheros: `which`, `whereis`, `locate`

Comando `which` muestra la localización de comandos

- Formato:

  `which [-a]` *`comando`*

- Opciones:
  - `-a` muestra todas las localizaciones del comando
- Ejemplo:

```
$ which ls
/bin/ls
```

Comando `whereis` muestra la localización del binario, fuente y página de manual de un comando

- Formato:

  `whereis [`*`opciones`*`]` *`comando`*

- Opciones:
  - `-b/-m/-s` muestra sólo el binario/manual/fuente
- Ejemplo:

```
$ whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
```

- Para más opciones ver la página de manual

Comando `locate` localiza ficheros rápidamente

- utiliza una base de datos donde guarda la localización de los ficheros (`/var/cache/locate/locatedb`)
- esa base de datos la crea y actualiza el administrador con el comando `updatedb`

- Ejemplo:

```
$ locate \*.bak
/root/.mozilla/firefox/2gwif.default/bookmarks.bak
/var/backups/group.bak
/var/backups/inetd.conf.bak
```

- Ver página de manual para opciones

2.4. Particiones y sistemas de ficheros

Vimos en el tema 2 como crear particiones y sistemas de ficheros en el momento de la instalación

- si añadimos un nuevo disco al sistema ya instalado deberemos crear las particiones y los sistemas de ficheros
- esta operación implica los siguientes pasos:
  1. creación de particiones (comando `fdisk`)
  2. creación de los sistemas de ficheros (comando `mkfs`)
  3. montado de los sistemas de ficheros (comando `mount`)

Creación de particiones

El comando para crear particiones es `fdisk`

- Formato:

  `fdisk [`*`opciones`*`] `*`dispositivo`*

  donde *`dispositivo`* es el dispositivo del disco (`/dev/hd`*`x`* en IDE, `/dev/sd`*`x`* para SCSI o SATA)

- Debemos tener permiso de administrador para usarlo
- Opciones:
  - `-l` muestra la tabla de particiones del dispositivo

`fdisk` se usa mediante un menú:

```
# fdisk /dev/hdb
The number of cylinders for this disk is set to 20805.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)
Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
```

Para crear una partición primaria de 5 GB usamos `n` (*new*):

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-20805, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-20805, default 20805): +5G

Command (m for help): p

Disk /dev/hdb: 10.7 GB, 10737418240 bytes
16 heads, 63 sectors/track, 20805 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
```

```
   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1               1        9689     4883224+  83  Linux
```

Por defecto, crea la partición tipo Linux (Id 83)

- con `l` (*list*) vemos el tipo de particiones soportadas
- para cambiar el tipo de partición se usa `t` (type)

```
Command (m for help): t 1
Selected partition 1
Hex code (type L to list codes): 82
Changed system type of partition 1 to 82 (Linux swap / Solaris)

Command (m for help): p

Disk /dev/hdb: 10.7 GB, 10737418240 bytes
16 heads, 63 sectors/track, 20805 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1               1        9689     4883224+  82  Linux swap / Solaris
```

Para que se guarden los cambios debemos usar `w` (*write*)

Otras herramientas `fdisk` escribe la tabla de particiones $\longrightarrow$ el contenido de los discos se pierde

Existen otras herramientas para modificar las particiones:

cfdisk interfaz para el `fdisk` (también escribe la tabla de particiones)

parted programa de GNU que permite crear, destruir, cambiar el tamaño, chequear y copiar particiones

qtparted programa Linux para manejar particiones, con interfaz gráfico (basado en `libparted`)

Creación de los sistemas de ficheros

Sobre cada partición debemos crear sistemas de ficheros con el comando `mkfs`

- Formato:

  `mkfs [-V] [-t` *`filesystem`*`] `*`dispositivo`*` [`*`n_bloques`*`]`

- Opciones:
  - `-V` verbose
  - `-t` *`filesystem`* tipo de sistema de ficheros a crear (ext2/3/4, xfs, etc.)
    - si no se especifica se crea el por defecto del sistema (en Linux ext2)
  - *`n_bloques`* número de bloques usados para el sistema de ficheros (si no se pone, se usa toda la partición)

`mkfs` es un *front-end* a distintos comandos, que permiten crear particiones de los tipos específicos:

- `mkfs.ext2` o `mke2fs` crea sistemas ext2
- `mkfs.ext3` crea sistemas ext3, equivalente a `mkfs.ext2 -j`
- `mkfs.jfs`, `mkfs.reiserfs`, `mkfs.xfs` crean sistemas JFS, ReiserFS y XFS, respectivamente
- `mkfs.msdos`, `mkfs.vfat` crea sistemas MS-DOS
- `mkswap` crea un sistema de ficheros de tipo Linux swap

Cada uno de estos comandos pueden tener distintas opciones

- ver las páginas de manual para más detalles

Comandos relacionados

- `dumpe2fs` muestra información de sistemas de ficheros ext2/3/4
  - información sobre inodos, bloques y grupos
- `tune2fs` permite ajustar parámetros en sistemas ext2/3/4
  - p.e. define el intervalo entre chequeos automáticos, convierte ext2 en ext3, etc.

- `e2label` cambia la etiqueta de un sistema ext2/3/4
- existen comandos similares para otros tipos de sistemas de ficheros, p.e. `reiserfstune`, `jfs_tune`, etc.

Partición de intercambio

- Si lo que creamos es una partición de intercambio, la debemos inicializar con `mkswap`

```
# fdisk -l /dev/hdb
Disk /dev/hdb: 10.7 GB, 10737418240 bytes
16 heads, 63 sectors/track, 20805 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdb1               1        9689     4883224+  83  Linux
/dev/hdb2            9690       20805     5602464   82  Linux swap / Solaris
# mkswap /dev/hdb2
Setting up swapspace version 1, size = 5736919 kB
no label, UUID=a6c2849b-c33e-478e-8a8d-fecfe3f18f6d
```

- Una vez creada debemos activarla con `swapon`

```
# swapon /dev/hdb2
# swapon -s
Filename                        Type            Size    Used    Priority
/dev/hda7                       partition       377488  0       -1
/dev/hdb2                       partition       5602456 0       -2
```

- Finalmente, para que se active en el arranque, debe incluirse la entrada correspondiente en el fichero `/etc/fstab`

```
/dev/hda2   none   swap    sw       0       0
```

Montado de los sistemas de ficheros

Para poder acceder a los sistemas de ficheros debemos *montarlos*

- los comandos para montar y desmontar son: `mount` y `umount`

Comando `mount` Permite asociar (*montar*) directorios a sistemas de ficheros

- Ejemplo

  ```
  $ mount -t ext3 /dev/hdb1 /home2
  ```

- Formato

  `mount [`*`opciones`*`] [-t `*`tipo`*`] [`*`disp.`*`] [`*`dir.`*`]`

- Algunas opciones:

  - *`tipo`* tipo de sistema de ficheros (ext2/3/4, reiserfs, vfat, etc.); si se pone `auto` intenta determinar de forma automágica
  - `-a` monta los filesystems listados en `/etc/fstab`
  - `-r`/`-w` monta los sistemas de sólo lectura/escritura
  - `-f` simulación; usado con `-v` (verbose) para chequear
  - `-n` monta sin añadir la entrada a `/etc/mtab`; se usa cuando `/etc` es sólo lectura
  - `-o` *`opciones`* opciones de montado; formato igual al usado en el fichero `fstab`

Comando `umount` Desmonta los sistemas de ficheros

- Formato

  `umount [`*`opciones] directorio`*

- Ejemplo:

  ```
  $ umount /home2
  ```

- Algunas opciones

  - `-a` desmonta los filesystems listados en `/etc/mtab`
  - `-r` si falla, intenta remontar sólo lectura
  - `-f` fuerza el desmontado

- Si hay algún proceso bloqueando el filesystem, este no se puede desmontar:

  - usar el comando `fuser -c` *`directorio`* para ver el PID del proceso

Fichero `/etc/fstab` Al iniciar el sistema se montan los filesystems listados en `/etc/fstab`

- cada línea del fichero tiene las siguientes columnas

  ```
  (file system) (mount point)   (tipo)  (opciones)       (dump)  (pass)
  ```

- Ejemplo:

  ```
  /dev/hda9       /home           ext3    defaults           0       2
  ```

- Alguna de las opciones son:

  - `rw` monta tipo lectura/escritura
  - `ro` sólo lectura
  - `auto`/`noauto` monta/no monta con `mount -a` (monta/no monta al inicio)
  - `exec`/`noexec` Permite/no permite la ejecución de ficheros binarios en la partición
  - `suid`/`nosuid` permite/no permite que los bits setuid y setgid tengan efecto
  - `dev`/`nodev` interpreta/no interpreta dispositivos de bloques o caracteres en el filesystem
  - `async` toda la I/O se realiza de forma asíncrona
  - `user` puede montarlo un usuario (y sólo el que lo monta puede desmontarlo); implica las opciones `noexec`, `nosuid` y `nodev`, a menos que se fuercen (p.e. `user,exec,suid,dev`)
  - `users` puede montarlo/desmontarlo un usuario y el que desmonta no tiene que ser el que lo montó; implica las mismas opciones que `user`
  - `defaults` selecciona opciones por defecto (`rw`, `suid`, `dev`, `exec`, `auto`, `nouser` y `async`)

- Filesystems específicos pueden tener opciones específicas:

  - ver el manual de `mount` para más detalles

- Si un directorio aparece listado en el `fstab` puede montarse sin especificar el dispositivo:

  ```
  $ mount /home
  ```

- Campos `dump` y `pass`

  dump lo usa el comando `dump` para determinar de que filesystems hacer copias de seguridad

  - valor 1 o 0 según si la partición va a tener un backup controlado por `dump` o no (normalmente no se usa)

  pass lo usa el comando `fsck` para determinar el orden en que se chequean los filesystems al iniciar el sistema

  - si 0, el filesystem no se chequea
  - si > 0, los filesystems se chequean en el orden indicado por los números
    - si varios tienen el mismo número, se chequean en paralelo (si es posible)
    - normalmente `/` tendrá 1 y el resto 2

Fichero `/etc/mtab` Contiene una lista de los filesystem que están montados en el sistema

- Ejemplo de fichero `/etc/mtab`

```
$ cat /etc/mtab
/dev/hda1 / ext3 rw,errors=remount-ro 0 0
proc /proc proc rw 0 0
devpts /dev/pts devpts rw,gid=5,mode=620 0 0
tmpfs /dev/shm tmpfs rw 0 0
/dev/hda9 /home ext3 rw 0 0
/dev/hda8 /tmp ext3 rw 0 0
/dev/hda5 /usr ext3 rw 0 0
/dev/hda6 /var ext3 rw 0 0
usbfs /proc/bus/usb usbfs rw 0 0
/dev/hdb1 /home2 ext2 rw,nodev 0 0
```

Autofs Sistema que permite montar los filesystems "bajo demanda"

- cuando se accede al directorio, este se monta
- se desmonta automáticamente después de un tiempo de inactividad (por defecto, 5 minutos)
- suele usarse para montar sistemas remotos con NFS

- Ficheros de configuración:
  - `/etc/auto.master` define los puntos de montado
    - por cada uno de los puntos definidos, se inicia un proceso `automount` usando los parámetros indicados
    - Ejemplo de `auto.master`:

```
/home      /etc/auto.home
/misc      /etc/auto.misc   --timeout 60
```

  - Los ficheros le indican al `automount` los filesystems a montar
  - Ejemplo de `auto.misc`

```
cdrom      -fstype=iso9660,ro  :/dev/cdrom
windoz     -fstype=vfat        :/dev/hda1 home     -rw,hard,intr
server:/export/home
```

  - esto monta el cdrom y la partición `/dev/hda1` en los directorios `/misc/cdrom` y `/misc/windoz`, y el directorio remoto `/export/home` del sistema `server` en `/misc/home`
- Para más información ver el manual de `autofs` y `automount`, el Autofs Automounter HOWTO o el Automount mini-Howto

Chequeo del sistema de ficheros

Periódicamente es necesario chequear los sistemas de ficheros

- el comando básico para chequeo y reparación es `fsck`

Al igual que `mkfs`, `fsck` es un front-end a comandos específicos para cada filesystem:

- `e2fsck`, `fsck.ext2` o `fsck.ext3` chequean sistemas ext2/ext3
- `fsck.jfs`, `fsck.reiserfs`, `fsck.xfs` para JFS, ReiserFS y XFS
- `fsck.msdos`, `fsck.vfat` para sistemas MS-DOS

Alguno de los errores que pueden aparecer se deben a:

- Varios ficheros que usan el mismo bloque
- Bloques marcados libres y ocupados simultáneamente
- Número de enlaces erróneo

- Nodos-i conteniendo información pero que no están en la entrada del directorio (la información se recupera en el directorio `lost+found` con el número de nodo-i)
- Entradas del directorio que apuntan a nodos-i ilegales o vacíos
- etc.

Algunas de las opciones de `fsck` son:

- `-t` *`filesystem`* tipo de filesystem a chequear
- `-A` chequea los filesystems listados en `/etc/fstab`
- `-N` no ejecuta; simplemente indica lo que haría
- `-R` usado con `-A` no chequea el filesystem raíz

Otras opciones dependen del filesystem particular

- ver las páginas de manual para cada caso

Otras utilidades

- `du`: muestra el espacio de disco usado por los ficheros y subdirectorios de un directorio
  - Formato:

    `du [`*`opciones`*`] [`*`directorio`*`]`
  - Algunas opciones:
    - `-a` muestra valores para ficheros y directorios (por defecto, solo muestra directorios)
    - `-b`, `-k` tamaños en bytes/KBytes
    - `-h` salida más legible
    - `-s` muestra sólo la ocupación total
  - Ejemplo:

```
$ du -sh /home /usr
1,2G    /home
2,3G    /usr
```

- `df`: muestra el espacio de disco usado y disponible de los sistemas de ficheros montados

- Formato:

  `df [`*`opciones`*`]`

- Algunas opciones:
  - `-a` muestra todos los filesystems (incluso los de tamaño 0)
  - `-h` salida más legible
  - `-i` da información sobre los inodos
  - `-l` sólo muestra filesystems locales
  - `-T` muestra el tipo de sistema de ficheros
- Ejemplo:

```
$ df -h
Filesystem          Tamaño Usado  Disp Uso% Montado en
/dev/hda1              67M   50M   13M  80% /
tmpfs                  63M     0   63M   0% /dev/shm
/dev/hda9             272M  8,1M  250M   4% /home
/dev/hda8              23M  1,1M   20M   5% /tmp
/dev/hda5             464M   90M  350M  21% /usr
/dev/hda6              74M   44M   27M  63% /var
```

2.5. Sistemas de ficheros con LVM

En el tema 2 vimos como crear un sistema LVM; algunas de sus ventajas son:

- LVM proporciona mucha más flexibilidad a la hora de distribuir el espacio disponible
- LVM permite mover y cambiar de tamaño los volúmenes creados bajo su control
- Existen varios beneficios inmediatos por usar LVM:
  - Es posible aumentar y decrecer los volúmenes en caliente: esto permite redistribuir el espacio en las particiones según nos sea necesario; también se puede dejar espacio sin asignar e ir asignándolo según vaya siendo necesario
  - Es posible añadir espacio de almacenamiento al sistema de volúmenes: si se añade un nuevo disco a la máquina se puede añadir este espacio a LVM y hacer crecer los volúmenes que contiene para aprovechar el nuevo espacio de almacenamiento

En este apartado veremos comandos para manejar sistemas LVM (nota: todos estos comandos tienen distintas opciones, ver páginas de manual)

- Información acerca de un grupo de volúmenes: `vgdisplay` o `vgs`

```
# vgdisplay GrupoVolumen
  --- Volume group ---
  VG Name               GrupoVolumen
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  7
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                6
  Open LV               6
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               14,84 GB
  PE Size               32,00 MB
  Total PE              475
  Alloc PE / Size       473 / 14,78 GB
  Free  PE / Size       2 / 64,00 MB
  VG UUID               N2NjFx-7ISe-J7hH-EX03-231w-XfbS-eCYfv0
```

- Información acerca de un volumen lógico: `lvdisplay` o `lvs`

```
# lvdisplay /dev/GrupoVolumen/homelv
  --- Logical volume ---
  LV Name                /dev/GrupoVolumen/homelv
  VG Name                GrupoVolumen
  LV UUID                dI6BqF-LAeG-3fO9-jXcr-vNde-f7iF-y9Oa2l
  LV Write Access        read/write
  LV Status              available
  # open                 1
  LV Size                1,00 GB
  Current LE             32
  Segments               1
  Allocation             inherit
  Read ahead sectors     0
  Block device           253:1
```

- Información acerca de un volumen físico: `pvdisplay` o `pvs`

```
# pvdisplay /dev/hda2
  --- Physical volume ---
  PV Name               /dev/hda2
  VG Name               GrupoVolumen
  PV Size               9,88 GB / not usable 0
  Allocatable           yes (but full)
  PE Size (KByte)       32768
  Total PE              316
  Free PE               0
  Allocated PE          316
  PV UUID               U6rMMw-5Z9U-fhBH-4R6G-reeJ-ZVha-K4xyHs
```

Manejar volúmenes físicos y grupos de volúmenes

- Creación de un volumen físico (PV), sobre una partición tipo Linux LVM (8e)

  ```
  # pvcreate /dev/hdc1
  ```

- Crear un grupo de volúmenes (VG), de nombre NuevoGrupo a partir de dos PVs

  ```
  # vgcreate NuevoGrupo /dev/hda2 /dev/hdc1
  ```

- Activar un grupo de volúmenes: es necesario hacer esto antes de usarlo, pero normalmente el sistema lo hace por nosotros en el arranque

  ```
  # vgchange -a y NuevoGrupo
  ```

- Borrar un VG (es necesario desactivarlo antes)

  ```
  # vgchange -a n NuevoGrupo
  # vgremove NuevoGrupo
  ```

- Añadir el PV `/dev/hdc1` a un VG ya creado

  ```
  # vgextend GrupoVolumen /dev/hdc1
  ```

- Quitar PVs de un VG

  ```
  # vgreduce NuevoGrupo /dev/hda2
  ```

Trabajar con volúmenes lógicos

- Crear un volumen lógico (LV) de nombre testlv en el VG NuevoGrupo con un tamaño de 4.20 GB

```
# lvcreate -L4.20G -n testlv NuevoGrupo
```

- Crear un volumen lógico con nombre otrotestlv, con 2 stripes (número de volúmenes físicos entre los que se reparte el LV)

```
# lvcreate -i2 -L1G -n otrotestlv vg
```

- Destruir un volumen lógico (hay que desmontarlo antes)

```
# umount /dev/NuevoGrupo/otrotestlv
# lvremove /dev/NuevoGrupo/otrotestlv
```

- Agrandar un LV; se puede especificar el nuevo tamaño en bytes (`-L`) o LEs (`-l`), o la diferencia

```
# lvextend -L12G /dev/GrupoVolumen/homelv
# lvextend -L+1G /dev/GrupoVolumen/tmplv
# lvextend -l+200 /dev/GrupoVolumen/tmplv
```

- Reducir un LV: `lvreduce` funciona igual que el `lvextend`

Una vez creados los volúmenes lógicos sólo resta crear los sistemas de ficheros y montarlos

- se hace de la forma habitual (`mkfs`, `mount`)

```
# mkfs.ext3 /dev/GrupoVolumen/homelv
# mount -t ext3 /dev/GrupoVolumen/homelv /home
```

Si hemos agrandado un LV debemos agrandar el filesystem: comando `fsadm`

- Chequea y redimensiona un sistema de ficheros
- Front-end para los comandos específicos de diferentes filesystems (ext2/3/4, ReiserFS, XFS)

Ejemplo: aumenta a 2G el tamaño del filesystem

```
# fsadm resize /dev/mapper/volgr-usr 2048M
```

Si no se especifica tamaño, aumenta al máximo posible
Otros comandos dependen de cada filesystem particular:

ext2/3/4 comando `resize2fs` (en Debian, paquete `e2fsprogs`): permite alargar o encoger un filesystem ext2/3/etx4[4]

[4]en kernel 2.6 y ext3/4 no es necesario desmontar el filesystem

```
# resize2fs /dev/GrupoVolumen/homelv
```

ReiserFS comando `resize_reiserfs` o `resizefs.reiserfs` (en Debian, paquete `reiserfsprogs`)

XFS el filesystem debe montarse que sea extensible; se usa el comando `xfs_growfs` (en Debian, paquete `xfsprogs`)

JFS como XFS el filesystem debe montarse que sea extensible; se especifica el tamaño en el proceso de remontado:

```
# mount -o remount,resize=1048576 /home
```

2.6. Manejo de discos cifrados

El cifrado que se usa en el proceso de instalación es un cifrado de disco completo (*Full disk encryption*, FDE)

- Se cifran todos los bits del disco o de la partición
- Es diferente del cifrado a nivel de sistema de ficheros (*Filesystem-level encryption*, FLE) en el que se cifra el contenido de los ficheros, no los metadatos (nombre del fichero, fechas de modificación, etc.).

El subsistema de cifrado FDE en Linux 2.6 es *dm-crypt*

- Parte de la infraestructura *device mapper*, utilizada también en LVM2 o RAID software, y puede colocarse por encima de estos
  - Permite cifrar discos completos, particiones, volúmenes lógicos o volúmenes RAI software
- Comando básico: `cryptsetup`
- Permite utilizar el estándar LUKS (*Linux Unified Key Setup*)
- Fichero `/etc/crypttab`: indica en el arranque como descifrar los discos

LUKS

Estándar para cifrado de disco en Linux

- Facilita la compatibilidad entre distribuciones
- Incluye soporte para múltiples claves
- Revocación de contraseña efectiva
- Uso mediante el comando *cryptsetup*, con *dm-crypt* como backend

cryptsetup

Comando básico para el cifrado de disco

- Puede cifrar con o sin LUKS
- LUKS no permite cifrado con clave aleatoria, generada a partir de `/dev/random` o `/dev/urandom`[5]

Ejemplos de uso:

1. Cifrar y activar una partición usando formato LUKS y una contraseña como clave (todos los datos se pierden y debemos reinicir el filesystem)

```
# Para mayor seguridad, desmonta y sobreescribe
# la partición (puede ser muy lento)
umount /dev/hda8
dd if=/dev/urandom of=/dev/hda8
# Formatea la partición, cifrando con LUKS
cryptsetup luksFormat /dev/hda8
# Activa la partición, en el
# dispositivo /dev/mapper/hda8_crypt
cryptsetup luksOpen /dev/hda8 hda8_crypt
# Reinicia el sistema de ficheros
mkfs -t fstipo /dev/mapper/hda8_crypt
# Obtiene el UUID luks
crypsetup luksUUID /dev/hda8
```

2. Cifrar una partición de swap usando `/dev/urandom` como clave aleatoria (sin usar LUKS)

```
cryptsetup create hda7_crypt /dev/hda7 --key-file /dev/urandom
```

3. Usar un fichero de clave para una partición cifrada inicialmente con contraseña, eliminando la contraseña inicial

```
 # Crea un fichero aleatorio para usar como clave
dd if=/dev/urandom of=/etc/keys/hda6.luks bs=1 count=4096
# Cambia los permisos del fichero (debe ser de root)
chown root.root /etc/keys/hda6.luks
```

[5] `/dev/random`: genera números aleatorios basándose en la entropía (ruido) del sistema; `/dev/urandom`: genera números pseudoaleatorios usando la entropía como semilla. `/dev/random` genera números de mayor calidad, pero es lento y puede bloquearse si la entropía del sistema es baja; `/dev/urandom` es un poco menos seguro, pero puede ser adecuado

```
chmod 700 /etc/keys
chmod 400 /etc/keys/hda6.luks
# Añade el fichero hda6.luks como clave
cryptsetup luksAddKey /dev/hda6 /etc/keys/hda6.luks
# Comprueba que existen dos claves (slots)
cryptsetup luksDump /dev/hda6
# Borra el slot 0 con la clave original
# (impide usar esa clave, hacerlo solo después de comprobar
que el fichero luks funciona como clave)
cryptsetup luksKillSlot /dev/hda6 0 --key-file /etc/keys/hda6.luks
# Comprueba que el slot se ha borrado
cryptsetup luksDump /dev/mapper/hda6_crypt
# Por último, modifica el fichero crypttab para indicar
que se use el fichero luks
```

4. Extiende el dispositivo cifrado `GV-homelv_crypt`, después de haber extendido el volumen lógico sobre el que está definido

```
# cryptsetup resize /dev/mapper/GV-homelv_crypt
```

Fichero /etc/crypttab

Especifica en el arranque como se deben descifrar los discos
Ejemplo:

```
#<cifrado>  <original> <fichero clave>      <opciones>
hda7_crypt  /dev/hda7  /dev/urandom         cipher=aes-cbc-essiv:sha256,size=256,swap
hda8_crypt  /dev/hda8  none                 luks
hda6_crypt  /dev/hda6  /etc/keys/hda6.luks  luks
```

Línea 1 área de swap con cifrado aleatorio

Línea 2 partición con cifrado LUKS; `none` indica que se pide una contraseña en el arranque

Línea 3 partición con cifrado LUKS y clave obtenida desde fichero

Para evitar problemas (posibles cambios en el nombre de las particiones), es preferible substituir el nombre del dispositivo original por su UUID obtenido usando `crypsetup luksUUID`

```
# cryptsetup luksUUID /dev/hda8
0e44dcc3-2b1f-4349-9fb3-db82b547acd1
# cat /etc/crypttab
.....
hda8_crypt UUID=0e44dcc3-2b1f-4349-9fb3-db82b547acd1 none luks
.....
```

3. Gestión de usuarios

Todo usuario de un sistema UNIX debe tener una cuenta para poder acceder

- Cuenta UNIX: colección de características lógicas que especifican quien es el usuario y lo que puede hacer en el sistema
- Estas características incluyen:
  - el nombre de usuario (*login* o *user name*)
  - la contraseña (*passwd*)
  - grupo o grupos a los que pertenece
  - un identificador numérico (UID)
  - un identificador numérico del grupo por defecto (GID)
  - un directorio *home*
  - un *login shell*
  - una colección de ficheros de inicio

Dentro de las cuentas asociadas a usuarios podemos encontrar diferentes tipos.

- cuentas normales de usuario
- cuenta del administrador (*root*)
- cuentas especiales de los servicios (*nobody*, *lp*, *bin*, etc.):
  - usadas por servicios internos del sistema
  - aumentan la seguridad, al permitir que servicios del sistema no se ejecuten como *root*

3.1. Ficheros de información de los usuarios

La información de los usuarios y grupos está incluida en los siguientes archivos:

- `/etc/passwd` mantiene la información principal de cada cuenta: nombre de usuario, UID, GID, *login shell*, directorio *home*, contraseña (en sistemas antiguos), ...
- `/etc/shadow` en sistemas actuales, fichero sin permiso de lectura que guarda las contraseñas encriptadas

- `/etc/group` información sobre los grupos definidos en el sistema. nombre del grupo, GID y miembros del mismo
- `/etc/gshadow` contraseñas para grupos (no suele usarse)

Fichero `/etc/passwd`

Ejemplo de líneas de `/etc/passwd`:

```
root:x:0:0:root:/root:/bin/bash
pepe:x:1002:1002:Pepe Pótamo,123,981234321,:/home/pepe:/bin/bash
```

donde se indican (si aparecen `::` seguidos, el campo está vacío):

- `pepe`: identificación de usuario en el sistema, que deberían tener las siguientes características
  - únicos en toda la organización (no sólo en la máquina local)
  - preferiblemente corto, en minúsculas y sin caracteres acentuados (para evitar problemas)
  - fácil de recordar
  - de formato fijo para todos los usuarios (p.e. nombre+apellido)
- `x`: contraseña encriptada
  - si aparece una `x` la contraseña está en el fichero `/etc/shadow`
- `1002`: UID número identificador del usuario
  - para usuarios normales, número entre 1000 y 32767 (o 65535 en sistemas actuales)
  - números por debajo de 1000 para usuarios especiales del sistema (`root` usualmente número 0)
  - el UID para un usuario debería ser único, y el mismo para todas las máquinas
  - se debe evitar reutilizar un UID, para evitar problemas de pertenencia de archivos
- `1002`: GID código del grupo principal al que pertenece el usuario
- `Pepe Pótamo,123,...`: información GECOS
  - cualquier cosa, usualmente el nombre completo del usuario y información adicional (n. de despacho, teléfono, etc.)

- `/home/pepe`: directorio personal del usuario
- `/bin/bash`: shell interactivo que utilizará el usuario

Fichero `/etc/shadow`

Fichero de acceso restringido que almacena las contraseñas encriptadas:

```
pepe:$1$.QKDPc5E$SWlkjRWexrXYgc98F.:12825:0:90:5:30:13096:
```

Contiene para cada usuario la contraseña encriptada y otros campos separados por :

- día, contado como número de días desde el 1/1/1970 (también conocido como *epoch*), en que la contraseña se cambió por última vez
  - si vale 0 se fuerza a que el usuario cambia su contraseña la primera vez que se conecta
- número de días antes de que pueda ser cambiada
- número de días de validez de la contraseña
- días en que se avisa al usuario de que la contraseña va a caducar
- días, una vez expirada, en que se deshabilitará la cuenta
- día, desde el 1/1/1970, en que la cuenta se inhabilitará
  - si no aparece nada, la cuenta no se inhabilita nunca
- un campo reservado

Fichero `/etc/group`

Información sobre los grupos de usuarios

```
users:x:100:pepe,elena
```

donde tenemos

- nombre del grupo
- contraseña del grupo (no suele usarse)
  - si `x`, se guarda en el fichero `/etc/gshadow`

`grupo:contraseña:administradores:miembros`

- los administradores pueden cambiar la contraseña, añadir usuarios al grupo, etc.

- la contraseña puede fijarse/cambiarse con el comando `gpasswd`

- GID identificador numérico del grupo

- lista de usuarios que pertenecen al grupo

Cambio de grupo

- un usuario puede cambiar de grupo con `newgrp`

  - si el grupo no tiene contraseña y no está en `gshadow` sólo pueden cambiar los miembros del grupo
  - si el grupo tiene contraseña, el usuario debe especificar la contraseña
  - si el grupo aparece en `gshadow`, la lista de miembros en este fichero pueden cambiar sin contraseña

Otros ficheros

Cuando se crea un nuevo usuario, los ficheros de inicio se copia del directorio `/etc/skel`

- el administrador debe crear unos ficheros adecuados para los usuarios, especificando los paths necesarios de ejecución, inicialización de variables del sistema, etc.

- también pueden usarse los ficheros `/etc/profile` o `/etc/bash.bashrc` (ver Tema 3, *Ficheros de inicialización de Bash*)

3.2. Creación manual de una cuenta

Implica los siguientes pasos.

1. Editar el fichero `/etc/passwd` y añadir una línea para la nueva cuenta

   - para evitar corrupción del fichero usar el comando `vipw`

2. Si usamos `shadow`, poner `x` en el campo de contraseña y editar el fichero `/etc/shadow`

   - para evitar corrupción del fichero usar el comando `vipw -s`

3. Editar `/etc/group` para añadir un nuevo grupo, si es necesario, o para añadir el usuario a los grupos que deseemos
    - si es necesario, editar `/etc/gshadow`
4. Crear el directorio del usuario
5. Copiar los ficheros de `/etc/skel` al directorio del usuario
6. Usar `chown`, `chgrp` y `chmod` para fijar el usuario, grupo y permisos del directorio
7. Fijar la contraseña con `passwd`
    - el usuario debería cambiar la contraseña tan pronto como fuera posible
    - puede forzarse modificando el fichero `shadow` o con la opción `-e`

Comando `passwd`

Permite fijar o cambiar la contraseña de un usuario. Opciones:

- `-e`, `--expire` fuerza a que la contraseña de la cuenta caduque; el usuario debe cambiarla en el siguiente login
- `-d`, `--delete` borra la contraseña
- `-l`/`-u`, `--lock`/`--unlock` bloquea/desbloquea la cuenta
- `-m`, `--mindays MIN_DAYS` número mínimo de días entre cambios de contraseña
- `-x`, `--maxdays DÍAS_MAX` número de días de validez de la contraseña
- `-w`, `--warndays DÍAS_AVISO` número de días de aviso de caducidad
- `-i`, `--inactive INACTIVO` días en que se deshabilitará la cuenta una vez expirada la contraseña
- `-S`, `--status` indica el estado de la contraseña (bloqueada `L`, sin contraseña `NP` o con contraseña válida `P`)

Eliminación manual de una cuenta

Implica los siguientes pasos

1. Inhabilitar la cuenta impidiendo el acceso
2. Hacer un backup de los ficheros de usuario
3. Eliminar los ficheros de usuario

La inhabilitación de la cuenta podría ser temporal o definitiva

- Temporal: puede hacerse de diversas formas
  - cambiar el login shell a `/bin/false`, o `/usr/sbin/nologin` (si disponible)
  - cambiar el campo contraseña de `/etc/passwd` a `*` (volviendo a poner `x` se habilita la cuenta),
  - poner una `!` al principio del campo contraseña de `/etc/shadow`,
  - usar `passwd -l`
- definitiva: borrar las entradas del usuario de `/etc/passwd` y `/etc/shadow`

3.3. Comandos para gestión de cuentas

Crear cuentas manualmente es un proceso tedioso:

- existen comandos que nos ayudan en la tarea

Comandos simples de manejo de cuentas

- `useradd` añade un nuevo usuario al sistema; ejemplo.

  ```
  useradd -c .Aitor Tilla.aitor
  ```

  - por defecto, sólo modifica los ficheros `passwd` y `shadow`, no crea el directorio home ni le pone contraseña (cuenta inhabilitada)
  - varias opciones:
    - `-m` crea el directorio home, si no existe (y copia los ficheros de /etc/skel)
    - `-g` *`grupo`* especifica el grupo principal
    - `-s` *`shell`* especifica la shell a utilizar

    - `-e` *`fecha`* fecha de expiración de la cuenta (formato YYYY-MM-DD)
  - Ejemplo:

    ```
    useradd -c .ªitor Tilla.ªitor -m -e 2006-11-02 -s
    /bin/bash -g staff
    ```

- `userdel` borra un usuario del sistema
- `usermod` modifica las cuentas de usuario
- `groupadd`, `groupdel`, `groupmod` lo mismo, para grupos
- `newusers` permite crear varias cuentas a partir de un fichero con nombres de usuario y contraseñas
  - las líneas del fichero deben tener el mismo formato que las del fichero `/etc/passwd`, con la contraseña sin encriptar
- `chpasswd` similar al anterior, permite actualizar las contraseñas de usuarios existentes:

  ```
  echo "pepe:pepepassword"| chpasswd
  ```

- `mkpasswd` obtiene la versión cifrada de una cadena para usar como contraseña:

  ```
  mkpasswd -m sha-512 mypassword
  ```

- `chsh` cambia el shell por defecto del usuario
- `chfn` cambia la información del campo GECOS

Comandos de alto nivel para el manejo de cuentas

- Comandos `adduser`, `addgroup`:
  - hacen de front-end de los de bajo nivel `useradd`, `groupadd` y `usermod`
  - crean los usuarios/grupos en función de la configuración especificada en el fichero `/etc/adduser.conf`
- Herramientas gráficas de gestión de usuarios y grupos:
  - `kuser` (KDE), `user-admin` (GNOME), etc.

Otros comandos relacionados

- `passwd`: permite cambiar la contraseña (ya comentado)
- `chage`: muestra y cambia la información de expiración de la contraseña
  - Formato:

    `chage [`*`opciones`*`] [`*`username`*`]`
  - Algunas opciones:
    - `-l` muestra información de expiración
- `su`: permite cambiar de usuario o pasar a ser administrador
  - Formato:

    `su [`*`opciones`*`] [-]` *`username`*
  - Si no se especifica el *`username`* pasa a administrador
  - Algunas opciones:
    - `-` inicia un login shell
    - `-m`, `-p` o `--preserve-environment` mantiene el entorno (no ejecuta el `.bashrc` del nuevo usuario)
    - `-s`, `--shell=`*`nueva_shell`* usa la shell especificada
    - `-c`, `--commmand=`*`comando`* ejecuta el comando con la identidad del nuevo usuario:

      ```
      su -c ´cat /etc/shadow´
      ```

3.4. Cuotas de disco

Algunos filesystems permiten limitar el uso del disco a los usuarios y grupos: **cuotas**

- Evitan que los usuarios monopolicen el disco
- Pueden causar problemas a los usuarios:
  - preferible instalar más disco o avisar a los usuarios que consuman demasiado

Límites de cuotas:

- **Límite débil**: si la cuenta del usuario o del grupo supera el límite débil, se impondrá un *período de gracia* en el que el usuario podrá reducir la ocupación

- **Límite duro**: se deniega cualquier intento de escribir datos después de este límite
- **Período de gracia**: tras superar el límite débil, si el usuario no resuelve el problema borrando archivos, la cuenta se bloquea

Cuotas de usuario y de grupos

- **Usuario**: fija un máximo al espacio de todos los ficheros del usuario
- **Grupo**: fija un máximo al espacio de todos los ficheros del grupo
  - puede incluir ficheros de varios usuarios

Instalación de cuotas de disco en Debian

Los pasos a seguir son:

1. Instalar el paquete `quota`
2. Modificar `/etc/fstab` para marcar los filesystems que tendrán cuotas:

   ```
   /dev/hda9  /home  ext4 defaults,usrjquota=aquota.user,grpjquota=aquota.group
   ```

3. Remontar el filesystem que hemos modificado

   ```
   mount -vo remount /home
   ```

4. Crear los índices de las cuotas (ficheros aquota)

   ```
   quotacheck -vguma
   ```

5. Activar las quotas:

   ```
   quotaon -va
   ```

6. Usar el comando `edquota` para editar las cuotas de usuarios y grupos

Comando `edquota`

Permite crear, manipular y eliminar cuotas basadas en usuarios o grupos

- Sintaxis:

  `edquota [`*opciones*`] [`*usuario*`|`*grupo*`]`

- Opciones:

- `-u` *`usuario`* configura las cuotas del usuario
- `-g` *`grupo`* configura las cuotas para un grupo
- `-f` *`filesystem`* realiza las operaciones obre un filesystem concreto (por defecto, lo hace sobre todos los filesystems que admitan cuotas)
- `-t` configura el período de gracia
- `-p` *`user1 usuarios`* copia la configuración de cuotas de *`user1`* a los usuarios indicados

- Al ejecutar `edquota` se abre el editor indicado en la variable `EDITOR` (por defecto, `vi`) para modificar las cuotas:
  - se muestran los bloques de 1K en uso, así como los límites *soft* y *hard* (también para i-nodos o ficheros)
    - si un límite está a 0 no se aplica
  - esta información se guarda en los ficheros `aquota.user` y `aquota.group` en el directorio base del filesystem

Otros comandos

Existen otros comandos para la gestión de las cuotas:

- `quotacheck` verifica la integridad de las bases de datos de las cuotas
  - se ejecuta en el script de inicio del sistema de cuotas
  - debe ejecutarse con las cuotas desactivadas
- `quotaon`/`quotaoff` activa/desactiva el sistema de cuotas
- `repquota` genera un informa del uso de las cuotas

```
# repquota /home
*** Report for user quotas on device /dev/hda9
Block grace time: 7days; Inode grace time: 7days
                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --   34920       0       0              6     0     0
tarabelo  --     728       0       0             31     0     0
tomas     *-     108     100     200  7days       8     0     0
```

- `quota` permite al usuario ver el estado de sus cuotas
  - Algunas opciones:
    - `-g` muestra información sobre las cuotas del grupo del usuario
    - `-v` imprime información incluso para los filesystem sin límite en la cuota
    - `-q` imprime un mensaje si se ha superado la cuota
  - Ejemplo

```
$ quota
Disk quotas for user tomas (uid 1001):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
      /dev/hda9     108*    100     200   6days       9       0       0
      /dev/hda8       1      10      20               1       0       0
```

- Para más información sobre la configuración de las cuotas ver Quota mini-HOWTO

4. Instalación y configuración básica de redes de área local

Linux soporta múltiples protocolos y hardware de red:

- Protocolos como TCP/IP y TCP/IP versión 6, IPX/SPX, PPP, SLIP, X.25, Frame Relay, etc.
- Soporta hardware para redes Ethernet, Token-Ring, etc
- Diferentes NICs (*Network Interface Cards*) implican diferentes dispositivos de comunicación:
  - `eth`*x* para Ethernet, `tr`*x* para Token-Ring, `pppx` para PPP, `sl`*x* para SLIP,
  - Además, existe el dispositivo de *loopback* `lo`
    - Funciona como un circuito cerrado en el que cualquier datagrama que se le pase como parámetro es inmediatamente devuelto a la capa de red del sistema
    - Se utiliza para realizar pruebas, y para un par de aplicaciones de red

- En muchos UNIX estos dispositivos aparecen en `/dev`
  - En Linux se crean dinámicamente por software y no requieren los ficheros de dispositivos
- En Linux puede ser necesario incluir los módulos adecuados para cada dispositivo

En esta sección trataremos la configuración de TCP/IP en redes Ethernet; para más información ver:

- Administración de red en Linux: Linux Network Administrators Guide 2 ed., Olaf Kirch y Terry Dawson
- Linux Networking-HOWTO
- Dispositivos de red soportados en Linux: Linux Hardware Compatibility HOWTO - Network adapters

4.1. Comandos de configuración de red

Los comandos más importantes para configurar la red son:

- `ifconfig`: configuración del interfaz de red
- `route`: configuración del routing
- `netstat`: información de la red

Comando `ifconfig`

Muestra y configura una interfaz de red:

```
$ /sbin/ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:12:43:A6:05:5C
          inet addr:193.144.84.77  Bcast:193.144.84.255  Mask:255.255.255.0
          inet6 addr: fe80::211:43ff:fea6:55c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1035446 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1053062 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:196973192 (187.8 MiB)  TX bytes:270128587 (257.6 MiB)
          Interrupt:169
```

- Sintaxis:

```
ifconfig [opciones] [interfaz]
ifconfig interfaz [configuración] [up|down]
```

- Opciones de visualización:
  - `-a` muestra todas las interfaces , incluso las inactivas
  - `-s` muestra información resumida (igual que `netstat -i`)
- En las opciones de configuración se indica entre otras cosas la IP, máscara de red y dirección de broadcast:

```
# ifconfig eth0 193.144.84.77 netmask 255.255.255.0 broadcast 193.144.84.255 up
```

- `ifconfig` permite también configurar el estado del interfaz, por ejemplo, cambiar el MTU, poner modo promiscuo, activar/desactivar ARP, cambiar su dirección hardware (si el dispositivo lo permite), etc.

```
# ifconfig eth0 mtu 500
# ifconfig eth0 -noarp
# ifconfig eth0 hw ether 52:54:00:12:34:56
```

- ver el manual de `ifconfig` para más información

Otros comandos relacionados

Otros comandos de configuración de interfaz son:

- `ifup/ifdown` activan/desactivan un interfaz de red

```
# ifdown eth0
```

- `iwconfig` configura un interfaz wireless

```
# iwconfig eth1 essid "Mi Red"
```

- `ip` muestra y modifica dispositivos y rutas
  - Alternativa a `ifconfig` y `route`
  - Más potente y complejo

Comando `route`

Permite modificar la tabla de routing, mostrando, añadiendo o borrando rutas

- muestra las rutas definidas
- permite añadir/borrar rutas estáticas
- permite definir un gateway de salida por defecto para conectarnos al exterior
- permite configurar el sistema para que actúe como un `router`

Mostrar una tabla de routing

Se usa `route [-n -e -ee]` (equivale a `netstat -r`)

```
$ /sbin/route -n -ee
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface    MSS   W
193.144.84.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0     0     0
0.0.0.0         193.144.84.1    0.0.0.0         UG    0      0        0 eth0     0     0
```

- Opciones:
  - `-n` usa direcciones IP en vez de nombres
  - `-e` emplea el mismo formato que `netstat -r`
  - `-ee` salida larga
- Los flags indican el estado de la ruta
  - `U` la interfaz está activa (Up)
  - `H` el destino es una estación (Host)
  - `G` la ruta usa una pasarela (Gateway)
  - `D` ruta creada dinámicamente por un demonio de encaminamiento o un mensaje ICMP de redirección
  - `M` ruta modificada dinámicamente
  - `R` ruta rehabilitada
  - `!` ruta rechazada
- De las siguientes columnas, algunas no se usan

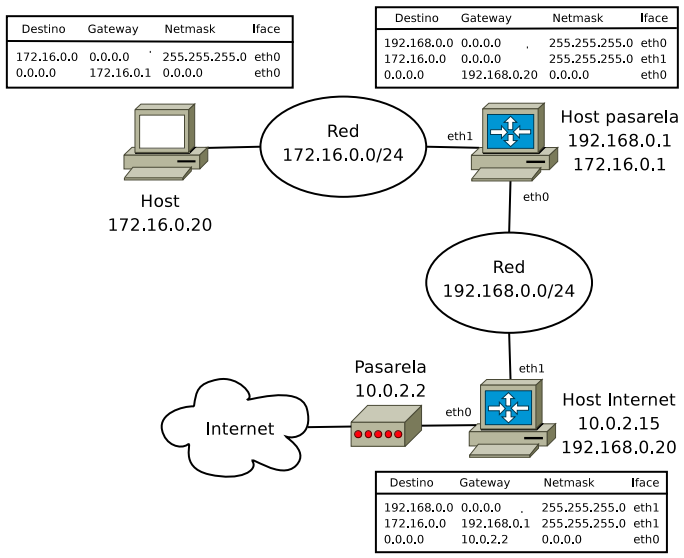
- `Metric` distancia (normalmente en saltos) al destino
- `Ref` número de referencias a la ruta (no usado en linux)
- `Use` número de consultas para la ruta
- `MSS` (*Maximum Segment Size*) tamaño máximo del segmento para las conexiones TCP en esa ruta
- `Window` Tamaño predeterminado de la ventana para las conexiones TCP en esa ruta
- `irtt` (*Initial Round Trip Time*) valor inicial del temporizador TCP

Añadir/borrar rutas estáticas

Se usa

```
route [add|del] [default] [-net|-host] target [netmask
Nm] [gw Gw] [opciones] [[dev] If]
```

Ejemplo: suponer que tenemos la configuración del dibujo, y queremos crear la tabla de rutas para el host Internet



- Añadir la ruta para la red 192.168.0.0/24 en `eth1`

```
route add -net 192.168.0.0 netmask 255.255.255.0 dev eth1
```

- Añadir la ruta por defecto

```
route add default gw 10.0.2.2
```

- Añadir una ruta para la red 172.16.0.0/24, usando como pasarela en host con IP 192.168.0.1

```
route add -net 172.16.0.0 netmask 255.255.255.0 gw 192.168.0.1
```

- El host pasarela tiene que permitir routing entre sus interfaces; pasa eso debemos activar el *ip_forward*:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Otras opciones de configuración

Linux permite otras opciones para configurar la red, como definir alias de IP o configurar opciones sobre el tráfico

Alias de IP

- Permite configurar múltiples direcciones IP a un único dispositivo de red
  - podemos soportar varias subredes IP en una misma Ethernet
  - los alias se indican como *dispositivo:número*
- Ejemplo:

```
# ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
# ifconfig eth0:0 192.168.10.1 netmask 255.255.255.0 up
# route add -net 192.168.10.0 netmask 255.255.255.0 eth0:0
```

Opciones del IP

Linux permite configurar diversas opciones sobre el tráfico IP

- los cambios pueden hacerse mediante el comando `sysctl`, escribiendo en los archivos del directorio `/proc/sys/net/ipv4` o de forma permanente en el fichero `/etc/sysctl.conf`
- algunos de estos estos archivos tienen un 0 (opción desactivada) o un 1 (opción activada)
- otros pueden tener un valor

- algunas de las opciones son:
  - `ip_forward` si 1 permite routing entre interfaces (por defecto 0)
  - `ip_default_ttl` el tiempo de vida por defecto de los paquetes (por defecto 64 ms)

Información de la red: comando `netstat`

`netstat` muestra las conexiones de red, tablas de routing y estadísticas de interfaz

- Formato:

  `netstat [`*tipo de información*`] [`*opciones*`]`

- Algunos tipos de información:
  - (nada) muestra la lista de sockets abiertos

```
$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp     0      0    jumilla.dec.usc.e:58946 aiff.usc.es:telnet      ESTABLISHED
tcp     0      0    jumilla.dec.usc.e:43658 ulla.dec.usc.es:1301    ESTABLISHED
tcp     0      0    jumilla.dec.usc.e:35346 sd.cesga.es:ssh         ESTABLISHED
tcp     0      0    jumilla.dec.usc.es:ssh  ulla.dec.usc.es:1688    ESTABLISHED
tcp     0      0    jumilla.dec.usc.es:ssh  teneguia.dec.usc.:35161 ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type   State    I-Node Path
unix  8      [ ]         DGRAM           15368    /dev/log
unix  2      [ ]         DGRAM           194110   @/org/kernel/udev/udevd
unix  2      [ ]         DGRAM           15671    @/var/run/hal/hotplug_socket
```

  - `--route`, `-r` muestra las tablas de rutas (igual que `route` admite los flags -n, -e y -ee)
  - `--interface`, `-i` muestra un resumen del estado de las interfaces de red (igual que `ifconfig -s`)

```
$ netstat -i
Kernel Interface table
Iface   MTU Met   RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0   1500 0   5110292      0      0      0  5011769      0      0      0 BMR
```

    - MTU (*Maximum Transmission Unit*): tamaño máximo del datagrama

    - Met: valor de la métrica para esa interfaz
    - RX/TX paquetes recibidos/transmitidos
    - OK/ERR/DRP/OVR paquetes transmitidos correctamente (OK), erróneos (ERR), descartados por falta de recursos (DRP, *drop*) y perdidos por desbordamiento (OVR, *overrun*)
    - Las banderas (*flags*) indican el estado del interfaz:
        - `B`: dirección de difusión activa
        - `L`: la interfaz es un dispositivo de bucle local (*loopback*)
        - `P`: se reciben todos los paquetes (modo promiscuo)
        - `O`: ARP desactivado para este interfaz
        - `M`: el interfaz permite multicast
        - `A`: el interfaz recibe todos los paquetes multicast en la red (*allmulti*)
        - `R`: la interfaz funciona (*running*)
        - `U`: la interfaz está activa (*up*)
    - Este estado puede cambiarse con `ifconfig`:

```
# ifconfig eth0 promisc # Modo promiscuo
# ifconfig eth0 -arp # Desactiva ARP
```

- `--statistics`, `-s` muestra estadísticas para cada protocolo de red

```
$ netstat -s
Ip:
  5001746 total packets received
  7479 forwarded
  0 incoming packets discarded
  4894721 incoming packets delivered
  ......
```

- Cada uno de los modos anteriores tienen diferentes opciones
- Algunas opciones válidas para varios tipos son:
    - `--numeric` o `-n` muestra información numérica en vez de nombres para IPs, puertos, etc.
    - `--continuous` o `-c` imprime la información solicitada de forma continua
    - `--extend` o `-e` muestra información extendida (con `-ee` aún más información)
- Para más información ver la página del manual

Otros comandos de red

Comando `arp`

- `arp` manipula la cache de ARP:
  - muestra la tabla ARP
  - borra entradas
  - añade entradas manualmente

Ejemplo:

```
# arp
Address                  HWtype  HWaddress           Flags Mask   Iface
almansa.dec.usc.es       ether   00:0D:56:6F:E6:90   C            eth0
193.144.84.1             ether   00:E0:63:93:26:E5   C            eth0
teneguia.dec.usc.es      ether   00:C0:4F:A1:5D:89   C            eth0
```

- Flag: `C` dirección completa, `M` dirección añadida manualmente

Algunas opciones:

- `-i` *`interfaz`* muestra las entradas para el interfaz indicado
- `-a` *`hostname`* muestra las entradas para el host especificado
- `-d` *`hostname`* borra las entradas para el host especificado
- `-s` *`hostname hw_addr`* añade manualmente una entrada para el host especificado con la dirección hardware indicada
- `-n` *`interfaz`* no hace traducción de IPs a nombres

Comando `ping`

- Muestra la disponibilidad de conexión y la velocidad de transmisión con un host remoto:

```
$ ping 193.144.84.1
PING 193.144.84.1 (193.144.84.1) 56(84) bytes of data.
64 bytes from 193.144.84.1: icmp_seq=1 ttl=255 time=0.420 ms
64 bytes from 193.144.84.1: icmp_seq=2 ttl=255 time=0.396 ms
64 bytes from 193.144.84.1: icmp_seq=3 ttl=255 time=0.368 ms
```

- `ping` envía paquetes ICMP (*ECHO_REQUEST*) al destino y espera respuesta, midiendo el RTT
- muchos firewalls bloquean el tráfico ICMP por lo que el `ping` no funciona

Algunas opciones:

- `-b` permite `ping` a una dirección de broadcast
- `-c` ***COUNT*** para después de enviar ***COUNT*** paquetes ECHO_REQUEST
- `s` ***packetsize*** especifica el número de bytes a enviar (por defecto 56)

Comando `traceroute`

- Muestra la ruta que sigue un paquete hasta llegar a destino

```
$ traceroute www.elpais.es
traceroute to a1749.g.akamai.net (130.206.192.32), 30 hops max, 40 byte packets
 1  rutfis (193.144.64.1)  1.070 ms  0.688 ms  0.927 ms
 2  * * *
 3  10.56.5.1 (10.56.5.1)  57.463 ms  2.021 ms  1.923 ms
 4  193.144.79.72 (193.144.79.72)  2.507 ms  16.280 ms  2.080 ms
 5  GE2-0-0.EB-Santiago0.red.rediris.es (130.206.204.21)  25.681 ms  2.068 ms  1.965 ms
 6  GAL.SO2-0-0.EB-IRIS4.red.rediris.es (130.206.240.33)  10.959 ms  10.665 ms  10.710 m
 7  130.206.220.59 (130.206.220.59)  20.277 ms  10.781 ms  10.470 ms
 8  a130-206-192-32.deploy.akamaitechnologies.com (130.206.192.32)  11.011 ms  23.482 ms
```

- `traceroute` utiliza el campo TTL de la cabecera IP para obtener respuestas ICMP TIME_EXCEEDED de los host por los que pasa el paquete (envía paquetes UDP)
- los sistemas pueden no enviar mensajes de tiempo excedido: aparecen *
- si los firewalls bloquean el tráfico ICMP no veremos nada
- otros programas similares:
  - `traceproto`: permite especificar el protocolo a usar (TCP, UDP, ICMP) y el puerto a tracear (por defecto 80)
  - `tcptraceroute`: envía paquetes TCP SYN para evitar problemas con firewalls

Comandos `host`, `dig`, `nslookup`

- Permiten obtener la dirección IP de un sistema a partir del nombre o viceversa:

```
$ host www.elpais.es
www.elpais.es is an alias for elpais.es.edgesuite.net.
elpais.es.edgesuite.net is an alias for a1749.g.akamai.net.
a1749.g.akamai.net has address 130.206.192.38
a1749.g.akamai.net has address 130.206.192.32
```

- `nslookup` está desaprobado (*deprecated*) y no se recomienda su uso

Comando `mii-tool`

- Permite ver y/o configurar el estado de la unidad MMI (*Media Independent Interface*) de la tarjeta de red
  - Ethernet usa MII para autonegociar la velocidad de enlace y el modo duplex

```
# mii-tool -v eth0
eth0: negotiated 100baseTx-FD flow-control, link ok
  product info: vendor 00:08:18, model 16 rev 0
  basic mode:   autonegotiation enabled
  basic status: autonegotiation complete, link ok
  capabilities: 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
  advertising:  100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD flow-contro
  link partner: 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD flow-contro
# mii-tool --force=100baseTx-HD eth0
# mii-tool  eth0
eth0: 100 Mbit, half duplex, link ok
```

- Comando parecido (más complejo): `ethtool`

4.2. Ficheros de configuración de red

La configuración mediante `ifconfig` y `route` no se mantiene al apagar el sistema:

- durante el proceso de arranque la red se inicia mediante la ejecución de scripts del `init.d`

- `/etc/init.d/networking` en Debian
- `/etc/init.d/network` en RedHat

- Estos scripts leen los ficheros de configuración de la red
- Fichero `/etc/network/interfaces` en Debian

```
auto eth0
iface eth0 inet static
   address 193.144.84.77
   netmask 255.255.255.0
   network 193.144.84.0
   broadcast 193.144.84.255
   gateway 193.144.84.1
   # dns-* options are implemented by the resolvconf package, if installed
   dns-nameservers 193.144.75.9
   dns-search dec.usc.es
   name Tarjeta de red Ethernet
```

- Fichero `/etc/sysconfig/network-scripts/ifcfg-eth`x en RedHat

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
IPADDR=193.144.84.186
NETMASK=255.255.255.0
GATEWAY=193.144.84.1
TYPE=Ethernet
```

Otros ficheros de configuración

Fichero `/etc/resolv.conf` especifica el dominio y los servidores DNS

- Ejemplo:

```
domain dec.usc.es
search dec.usc.es usc.es
nameserver 193.144.75.9
nameserver 193.144.75.12
```

- si buscamos por un hostname (sin dominio) le añade `dec.usc.es` y si no aparece busca por `usc.es`
- pueden añadirse hasta tres servidores de DNS

Fichero `/etc/hosts` fichero que asocia nombres de hosts con direcciones IP

- permite consultar una IP sin acceder al DNS
- Ejemplo de `/etc/hosts`:

```
127.0.0.1     localhost.localdomain localhost
193.144.84.77 jumilla.dec.usc.es jumilla
```

- la consulta es más rápida que acceder al DNS
  - si las IPs cambian la dirección es incorrecta
- sólo debería aparecer el nodo local y la interfaz de loopback
  - esto permite fijar el nombre y el dominio del sistema
  - en algunas distribuciones (Debian) el nombre también debe ponerse en el fichero `/etc/hostname`
  - el nombre y el dominio pueden obtenerse mediante los comandos `hostname` y `dnsdomainname`

Fichero `/etc/networks` fichero de texto que asocia nombres a redes

- No es imprescindible
- Ejemplo de `/etc/networks`

```
red1 172.16.1.0
red2 172.16.2.0
```

Fichero `/etc/host.conf` configura el comportamiento del *name resolver*

- indica donde se resuelven primero la dirección o el nombre de un nodo
- Ejemplo de `/etc/host.conf`:

```
order hosts,bind
multi on
```

- indica que primero se verifiquen las tablas locales (`/etc/hosts`) y después el DNS
- `multi on` indica que se retornen todas las direcciones válidas que se encuentren en `/etc/hosts`

Fichero `/etc/nsswitch.conf` fichero de configuración del *Name Service Switch*

- centraliza la información de diferentes servicios para la resolución de nombres
  - indica las acciones a realizar para acceder a las diferentes bases de datos del sistema: hosts, contraseñas, servicios, etc.
  - reemplaza la funcionalidad del archivo `host.conf`
  - introducido en la versión 2 de la biblioteca GNU
- Ejemplo de `nsswitch.conf`

```
hosts:          dns files
networks:       files
```

  - indica que un host se busque primero en el DNS y después en el fichero `/etc/hosts`, mientras que una red se busca sólo en `/etc/networks`
- Es posible controlar el comportamiento por medio de acciones, por ejemplo:

```
hosts:          dns [!UNAVAIL=return] files
networks:       files
```

- si el estado de salida del DNS es diferente de *no disponible* no consulta a los ficheros:
  - sólo accede a `/etc/hosts` si el DNS no está disponible
- Los valores de estado disponibles son:
  - `success` la petición se encontró sin errores (acción por defecto `return`)
  - `notfound` no error, pero no se encontró el nodo o la red (acción por defecto `continue`)
  - `unavail` servicio solicitado no disponible (acción por defecto `continue`)
  - `tryagain` servicio no disponible temporalmente (acción por defecto `continue`)

Fichero `/etc/protocols` lista los protocolos que reconoce el sistema operativo

- Ejemplo de `/etc/protocols`

```
ip      0       IP    # internet protocol, pseudo protocol number
icmp    1       ICMP  # internet control message protocol
tcp     6       TCP   # transmission control protocol
udp     17      UDP   # user datagram protocol
.....
```

Fichero `/etc/services` relaciona las aplicaciones con sus correspondientes puertos y protocolos básicos

- Un trozo de `/etc/services`

```
ftp-data        20/tcp   # Datos de ftp
ftp             21/tcp   # Control de ftp
ssh             22/tcp   # SSH por TCP
ssh             22/udp   # SSH por UDP
telnet          23/tcp   # Telnet
smtp            25/tcp   # Correo electrónico
....
```

4.3. Configuración del DHCP

DHCP (*Dynamic Host Configuration Protocol*) permite configurar automáticamente la red de los sistemas a partir de un servidor DHCP

- La información de IPs, DNS, etc. se mantiene centralizada en el servidor
- Al iniciarse, los clientes se conectan al servidor (por broadcast) y cargan su configuración

Configuración del servidor

Se encuentra en el fichero `/etc/dhcp/dhcpd.conf`

- Ejemplo sencillo de configuración

```
option domain-name "midominio.com";  # Nombre de Dominio
option domain-name-servers 10.0.2.3, 193.14.75.9; # Servidores de Nombres
default-lease-time 600; # Tiempo por defecto que dura una asignación
```

```
max-lease-time 7200;    # Duración máxima de una asignación
option subnet-mask 255.255.255.0; # Máscara de red
subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.10 192.168.0.20;
  option broadcast-address 192.168.0.255; # Dirección de Broadcast
  option routers 192.168.0.1;             # Gateway de la red
}
host marte {
  hardware ethernet 52:54:00:12:34:70;
  fixed-address marte.mired.com;
}
```

- si utilizamos nombres (como `marte.mired.com`) la IP debe ser accesible (por DNS o `/etc/hosts`)
- en el fichero `/etc/default/isc-dhcp-server` debemos especificar el interfaz por el que servimos DHCP
- en `/var/lib/dhcp/dhcpd.leases` están las IPs asignadas
- para más información ver la página de manual de `dhcpd`

Configuración del cliente

Para que el cliente obtenga los datos de DHCP usar:

```
# dhclient eth0
```

- Un comando similar es `pump`

Para que el cliente se configure en el inicio debemos modificar el fichero de configuración de red

- En Debian, fichero `/etc/network/interfaces`:

```
auto eth0
iface eth0 inet dhcp
```

5. Automatización de tareas

En esta sección veremos la utilización de comandos que permiten automatizar tareas repetitivas

1. Tareas que se deben ejecutar en momentos concretos o de forma periódica:
   - `at`, `batch` permiten ejecutar trabajos a una hora específica o bajo determinadas condiciones
   - `cron` permite correr trabajos a intervalos regulares
2. Herramientas para automatizar la configuración de servidores

5.1. Tareas periódicas

Comando `at`

Permite indicar el momento en que se quiere ejecutar un trabajo

- Sintaxis:

  `at [`*`opciones`*`]` *TIME*

- Al ejecutar `at` pasamos a un nuevo prompt, que nos permite introducir comandos que se ejecutarán a la hora indicada
  - para salvar el trabajo y salir `CTRL-D`
  - el entorno actual también se salva
  - al terminar, la salida estándar se envía como un mail al usuario
  - el trabajo no se para al salir de la sesión
- Ejemplo:

```
$ at 11:45
warning: commands will be executed using /bin/sh
at> ls /tmp > lista
at> <EOT>
job 4 at Wed Nov 16 11:45:00 2005
```

- Algunas opciones:
  - `-f` *`FILE`* especifica un fichero conteniendo las acciones a realizar en vez de la entrada estándar
  - `-c` *`jobnumber`* muestra el trabajo a ejecutar
  - `-m` envía un mail al usuario, incluso aunque no haya salida
  - `-v` muestra la hora a la que se ejecutará el trabajo

- *TIME* puede especificarse de varias formas:
  - `HH:MM` por ejemplo `12:54`
  - `HH:MMAM/PM`, por ejemplo `1:35PM`
  - `HH:MM MMDDYY`, por ejemplo `1:35PM 122505`
  - `now + numero unidades`, donde *unidades* puede ser `minutes`, `hours`, `days`, o `weeks`

    ```
    $ at now+2hours
    ```
  - `today`, `tomorrow`, por ejemplo `12:44tomorrow`
  - `midnight` (00:00), `noon` (12:00), `teatime` (16:00)

Comandos relacionados

- `atq` o `at -l` lista los trabajos pendientes del usuario
  - si es el superusuario, lista los trabajos de todos los usuarios
- `atrm` o `at -d` borra trabajos identificados por su número de trabajo
- `batch` ejecuta trabajos cuando la carga del sistema es baja
  - el trabajo empieza en cuanto la carga caiga por debajo de 1.5
  - la carga se obtiene del fichero `/proc/loadavg`

Ficheros de configuración

- El administrador puede controlar la utilización de `at`
  - Ficheros `/etc/at.allow` y `/etc/at.deny`
    - `at.allow` lista los usuarios que pueden usar `at`
    - `at.deny` lista los usuarios que NO pueden usar `at`
- Primero se chequea `/etc/at.allow`
  - si está el usuario, puede usar `at`
  - si no está o el fichero no existe, se chequea `/etc/at.deny`
  - si el usuario no está en `at.deny` puede usar `at`
- Si no existe ninguno de los dos ficheros, solo root puede ejecutar `at`
- Para dar permiso para todos los usuarios crear sólo el fichero `at.deny` vacío

Procesos periódicos

Para crear trabajos que se ejecuten periódicamente se utilizan el demonio `cron` y el comando `crontab`

- `crontab` permite configurar los procesos periódicos
- `cron` se encarga de su ejecución

La utilización de `cron` se gestiona a través de los ficheros `/etc/cron.allow` y `/etc/cron.deny`

- el comportamiento si no existen los ficheros depende de la configuración del sistema
- en Debian, si no existen, todos los usuarios pueden usar `crontab`

Los trabajos se especifican en un fichero de `crontab`, que se guarda en `/var/spool/cron/crontabs`, y que puede tener tres tipos de líneas:

- Comentarios, que empiezan por `#`
- Definición de variables, de tipo *`nombre = valor`*

```
# shell usada para ejecutar los comandos
SHELL=/bin/bash
# Usuario al que se envía (por mail) la salida
# de los comandos (por defecto, se envían
# al propietario del fichero crontab)
MAILTO=pepe
```

- Especificación del trabajo y de la hora de ejecución, de la siguiente forma:

  `minuto hora día mes día_semana comando`

  - el día de la semana de 0 a 7 (0 ó 7 domingo)
  - `*` indica cualquier valor
  - se pueden indicar rangos, listas o repeticiones:
    - `1-5` para indicar de lunes a viernes
    - `0,15,30,45` para indicar cada 15 minutos
    - `0-23/2` en el campo hora, indica realizar cada dos horas (0, 2, 4, etc.)

- Ejemplos:
  - Borra el `/tmp` todos los días laborables a las 4:30 am

    ```
    30 4 * * 1-5 rm -rf /tmp/*
    ```
  - Escribe la hora, cada 15 minutos, durante la noche:

```
0,15,30,45 0-8,20-23 * * * echo Hora:$(date)»/tmp/horas
```

Comando `crontab`: crear y editar los trabajos periódicos

- Sintaxis:

  `crontab [-u` *`usuario`*`] {-l|-e|-r}`
  `crontab [-u` *`usuario`*`]` *`fichero`*

- en la segunda forma instala un nuevo `crontab` desde un fichero
- Opciones:
  - `-u` *`usuario`* crea o maneja el crontab de un usuario específico (sólo root)
  - `-e` edita el fichero `crontab`
  - `-l` muestra el fichero `crontab`
  - `-r` borra el fichero `crontab`

El demonio `cron` busca ficheros en `/var/spool/cron` para ejecutarlos a la hora indicada

- además también ejecuta las acciones indicadas en los ficheros `/etc/crontab` y en el directorio `/etc/cron.d/`
- estos ficheros suelen ser de mantenimiento del sistema

De esta forma, el administrador puede crear scripts que se ejecuten con periodicidad horaria, diaria, semanal y mensual

- sólo tiene que colocar esos scripts en los directorios `/etc/cron.hourly`, `/etc/cron.daily`, `/etc/cron.weekly` o `/etc/cron.monthly`
- la fecha y hora de ejecución de estos scripts se controla en el fichero `/etc/crontab`

Cron está pensado para sistemas funcionando 24/7

- Si el sistema está apagado a la hora de una acción cron, esa iteración no se realiza
- Problema en sistemas de sobremesa y/o domésticos

Solución complementaria: **Anacron**

Anacron

Ejecuta asíncronamente tareas periódicas programadas

- Al iniciarse el sistema comprueba si hay tareas periodicas pendientes (que no se realizaron por estar el sistema apagado)
- De ser así, espera un cierto retardo y las ejecuta

Fichero de configuración: `/etc/anacrontab`

período retardo id_del_trabajo comando

- El período se especifica en días (o como `@monthly`) y el retardo en minutos:

```
1        5   cron.daily   nice run-parts /etc/cron.daily
7        10  cron.weekly  nice run-parts /etc/cron.weekly
@monthly 15  cron.monthly nice run-parts /etc/cron.monthly
```

- Cada día, si el trabajo `cron.daily` no se realizo a la hora fijada por cron, se realiza 5 minutos después de iniciar el anacron

Limitaciones de anacron

- Solo para uso del administrador
- Solo permite períodos de días (no vale para tareas que se deban ejecutar varias veces al día)

5.2. Automatización de la configuración

Programas que permiten automatizar la gestión de la configuración de servidores en redes complejas

- La configuración se indica de forma centralizada
- Los diferentes servidores se configuran según lo indicado
- Liberan al administrador de tener que configurar manualmente los diferentes sistemas

Ejemplos más populares (open-source):

- Puppet: utiliza un lenguaje declarativo para describir la configuración de los sistemas
- Chef: Utiliza un Ruby para escribir "recetas" con la configuración de los sistemas

Una comparativa en Wikipedia

Puppet

Puppet usa un paradigma cliente-servidor

- En el servidor se describe la configuración de los sistemas (servicios, usuarios, ficheros,...) a través de ficheros de manifiesto
  - Se usa un lenguaje declarativo propio
  - Los clientes se actualizan periódicamente (30 minutos, por defecto)
- Compatible con múltiples SOs
  - La información concreta de los sistemas la obtiene mediante la utilidad *Facter*
  - Los manifiestos se compilan en un catálogo que contiene los recursos y sus dependencias

Más información: docs.puppetlabs.com
Ejemplo de manifiesto (fichero `site.pp`)

```
class ntp {
    package { "ntp":
        ensure => installed,
    }
    service { "ntp":
        ensure => running,
    }
}

node "cliente.midominio.com" {
    include ntp
    user { "pepe":
        ensure => present,
        uid => "1001", gid => "admin",
        shell => "/bin/bash", home => "/home/pepe",
        managehome => true,
    }
}
```

6. Copias de seguridad

Realizar copias de seguridad es una de las tareas más importantes del administrador del sistema

- Es casi inevitable que se produzcan pérdidas de información, debido a, entre otras causas:
  - deterioro o borrado accidental por parte de un usuario autorizado
  - ataque intencionado por parte de personas no autorizadas
  - fallo del software o el hardware
  - incendio, robos, y desastres naturales, etc.
- es imprescindible poder recuperar la información perdida

En esta sección veremos los comandos básicos para realizar copias de seguridad en UNIX/Linux; para más información:

- Backup & Recovery, W. Curtis Preston, O'Reilly, 2007
- Linux System Administrators Guide: Capítulo 12, Backups

6.1. Estrategias para las copias de seguridad

Una buena estrategia para copias de seguridad debe tener las siguientes características:

- Ser fácil de usar, preferiblemente si totalmente automática
- Eficiencia y rapidez:
  - compromiso entre el tiempo de backup y el tiempo de recuperación
- Facilidad de restauración
- Capacidad de verificar las copias
  - difícil si el sistema está siendo usado continuamente
- Tolerancia a fallos en los medias de almacenamiento (cintas, etc.)
  - necesidad de mantener al menos dos copias de los backups completos del sistema
  - al menos una de las copias debe almacenarse en otro sitio
- Portabilidad
  - posibilidad de recuperar la información en diferentes sistemas

Componentes de las copias de seguridad

Hay básicamente tres componentes que intervienen en una copia de seguridad:

- El planificador: decide que información se copia y cuando
- El programa de copia: los comandos que mueven los datos de los discos a los medios
- Los medios de almacenamiento: cintas, CDs, etc.

El planificador: Decide cuando realizar el backup y cuanta información copiar

- normalmente es gestionado mediante `cron`

Según la información que salvemos podemos hablar de los siguientes tipos de backup:

Completo se salva toda la información del sistema

Parcial sólo se salva la información más importante y difícil de recuperar

- los ficheros de usuario,
- los ficheros de configuración, p.e. `/etc/passwd`
- directorios de correo, web, etc.

Incremental sólo se salvan los ficheros modificados desde el último backup completo o incremental

- la copia de seguridad necesita menos tiempo y espacio
- para restaurar los datos necesitaremos el último backup completo y todos los incrementales

Diferencial se salvan los ficheros modificados desde el último backup completo

- los backups son más grandes que en el caso incremental
- para restaurar sólo necesitamos el backup completo y el último diferencial

Programa de copia: Se encarga de copiar los ficheros seleccionados en el medio de almacenamiento; dos mecanismos básicos

- Basado en imagen: accede al disco a bajo nivel
  - normalmente copias más rápidas, pero mas lento restaurar ficheros individuales
  - programas específicos para diferentes filesystems
  - comandos de este tipo son `dump` y `dd`
- Fichero a fichero
  - acceden a los ficheros a través de llamadas al SO
  - copias más lentas, pero restauración de ficheros individuales más simple
  - comandos de este tipo son `tar`, `cpio` o `afio`

Medios de almacenamiento: Dispositivos donde se guarda la información; los más populares son:

- Cintas, desde cintas DDS/DAT de 8 o 3,8mm con capacidades hasta unos 160 GB, hasta cartuchos de varios TBs (p.e. 8.5 TB en StorageTek T10000D)
  - Dos dispositivos: cinta con no-rebobinado y con rebobinado (en Linux para cintas SCSI `/dev/nst`*X* y `/dev/st`*X* respectivamente, en Solaris `/dev/rmt/`*X* y `/dev/rmt/`*X*)
  - Las cintas pueden controlarse a través del comando `mt`
- Discos duros externos
- Discos ópticos (CDs, DVDs)
  - necesitan software adicional como `mkisofs` (crea imágenes ISO) y `cdrecord` (graba en CDs o DVDs)
  - buenos cuando la cantidad de datos no es excesivamente elevada

6.2. Comandos básicos

Veremos los comandos básicos para hacer backups en UNIX: `tar`, `cpio` y `dump`

Comandos `dump` y `restore`

Comandos más comunes para copias de seguridad

- comandos originales de BSD UNIX
- dependen del tipo de *filesystem*

Comando `dump`: Hace copias de un sistema de archivos entero, con las siguientes características:

- Pueden ser copias multivolumen
- Puede salvar ficheros de cualquier tipo (incluido ficheros de dispositivos)
- Los permisos, propietarios y fechas de modificación son preservados
- Puede realizar copias incrementales
- También puede usarse para salvar ficheros individuales (no es lo usual)

El formato y los argumentos de `dump` dependen de la versión utilizada, pero en general es:

`dump [-nivel] [`*`opciones`*`] [`*`ficheros_a_salvar`*`]`

- Nivel de `dump`: entero entro 0-9:
  - 0 implica backup completo
  - mayor que 0 implica copiar sólo los ficheros nuevos o modificados desde el último backup de nivel inferior
  - `dump` guarda información sobre los backups realizados en el fichero `/etc/dumpdates` o `/var/lib/dumpdates`
- Algunas opciones:
  - `-f` especifica el dispositivo o fichero donde salvar la copia
  - `-u` actualiza el fichero `dumpdates` después de una copia correcta
  - `-a` determina automáticamente el fin de la cinta (opción por defecto)
  - `-j`, `-z` usa compresión con bzlib o zlib (sólo en algunas versiones)
- Ejemplo: backup de nivel 0 de la partición `/home`

```
# dump -0u -f /dev/st0 /home
```

- Ejemplo: backup en una máquina remota usando `ssh` como transporte

```
# export RSH=ssh
# dump -0u -f sistema_remoto:/dev/st0 /home
```

Comando `restore`: Restaura ficheros salvados por `dump`

- Formato:

```
restore acción [opciones] [ficheros_a_recuperar]
```

- Acciones principales:
  - `r` restaura la copia completa
  - `t` muestra los contenidos de la copia
  - `x` extrae sólo los ficheros indicados
  - `i` modo interactivo
    - permite ver los ficheros de la copia
    - con `add` indicamos los ficheros a extraer y con `extract` los extraemos
    - usar `?` para ayuda
- Algunas opciones:
  - `-f` especifica el dispositivo o fichero de la copia
  - `-a` no pregunta de que volumen extraer los ficheros (lee todos los volúmenes empezando en 1)
- Ejemplo: restaurar el backup de `/dev/st0`

```
# restore -rf /dev/st0
```

- Ejemplo: restaurar el backup desde un sistema remoto

```
# export RSH=ssh
# restore -rf sistema_remoto:/dev/st0
```

- Ejemplo: restaurar sólo un fichero

```
# restore -xaf /dev/st0 fichero
```

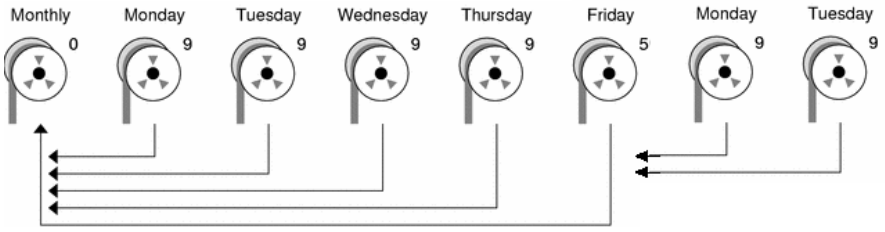
Archivo `restoresymtable`: Se crea cuando se restaura un filesystem completo, en el directorio donde se restaura

- Contiene información sobre el sistema restaurado
- Puede eliminarse una vez finalizada la restauración

Planificación de los backups

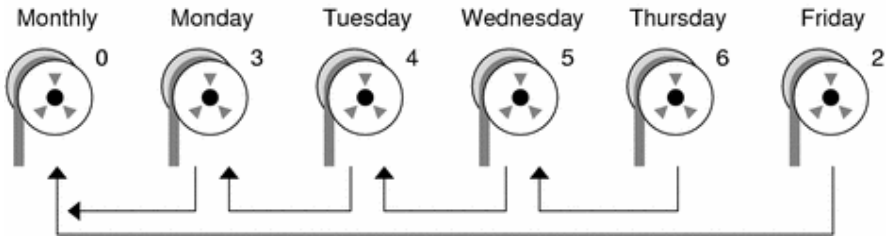
Podemos seguir diferentes estrategias a la hora de planificar los backups

- Ejemplo 1: copia de nivel 0 mensual, de nivel 9 diaria y de nivel 5 semanal



  - necesita 6 o 9 cintas: una para el 0, 4 para los niveles 5 y 1 o 4 para los niveles 9
  - para restaurar necesitamos restaurar en orden:
    1. la copia de nivel 0
    2. la última copia de nivel 5, y
    3. la última de nivel 9, después de la de nivel 5

- Ejemplo 2: copia de nivel 0 mensual, de nivel 2 semanal y de niveles 3, 4, 5 y 6 cada día



  - necesita al menos 9 cintas
  - para restaurar necesitamos restaurar en orden:
    1. la del nivel 0
    2. la del último viernes (nivel 2)
    3. las diarias desde el último viernes de forma consecutiva

Comando `tar` (Tape ARchiver)

Permite almacenar varios ficheros en uno sólo, manteniendo la estructura de directorios:

- Sintaxis:

  `tar [-]`*`función`*`[`*`modificador`*`] `*`fichero`*` [`*`directorio`*`]`

- Ejemplo: crea un fichero tar conteniendo los ficheros del directorio `/etc`

  ```
  tar cvf copia.tar /etc
  ```

- Puede indicarse un fichero o un dispositivo (p.e. `/dev/fd0`)
- `tar` conserva las propiedades de los ficheros: permisos, usuario/grupo, fechas, etc.
- Funciones principales:
  - `c` crea un nuevo archivo `tar`
  - `x` extrae los ficheros del archivo
  - `t` lista los ficheros del archivo
  - `r` añade nuevos ficheros al final del archivo `tar`
  - `u` almacena sólo los ficheros nuevos o modificados respecto a los del archivo `tar`
  - `A` añade un fichero `tar` a otro
  - `d` obtiene las diferencias entre los ficheros de la copia y los del disco
  - `--delete` borra un fichero del archivo `tar`
- Algunas opciones:
  - `v` verbose, muestra lo que está haciendo
  - `f` para indicar el nombre del fichero `tar`; por defecto toma `-` que representa la entrada/salida estándar
  - `z` comprime la copia con `gzip`
  - `--bzip2` o `j` comprime la copia con `bzip2`
  - `l` almacena sólo los ficheros locales (útil con NFS)
  - `k` no sobreescribe los ficheros existentes al extraer

    - `T` o `--files-from F` obtiene la lista de ficheros a guardar del fichero `F`
    - `X` o `--exclude-from=F` excluye los ficheros que concuerdan con los patrones listados en el fichero `F`
    - `N` o `--newer DATE` sólo guarda los ficheros más nuevos que `DATE`
    - `M` o `--multi-volume` permite crear copias multivolumen (por ejemplo, varios disquetes)
- para más opciones ver la página de info (`info tar`)
- Ejemplos:
    - Extrae todos los ficheros de `copia.tar`

      ```
      tar xvf copia.tar
      ```
    - Extrae el el fichero `passwd` de `copia.tar`

      ```
      tar xvf copia.tar etc/passwd
      ```
    - Copia el contenido de `/tmp` directamente a un disquete

      ```
      tar cvf /dev/fd0 /tmp
      ```
    - Copia un directorio completo

      ```
      (cd dir1 && tar cf - .) | (cd dir2 && tar xvf -)
      ```
    - Copia los ficheros más nuevos que un fichero `control`

      ```
      find dir -newer control ! -type d -print | tar cvfT
      f.tar -
      ```
- Problemas con `tar`
    - Algunas versiones no admiten opciones como la compresión o copia multivolumen
    - Algunas versiones tienen problemas con paths muy largos (más de 100 caracteres)

Comando `cpio`

El comando `cpio` es similar a `tar` en funcionalidad

- crea y extrae archivos, o copia ficheros de un lugar a otro
- maneja archivos en formato `cpio` y formato `tar`

Tres funciones primarias:

1. *Copy-out*: copia ficheros a un archivo, con la opción `-o`
   - Ejemplo: copia todos los directorios desde el actual en el fichero `tree.cpio`

     ```
     $ find . | cpio -ov > tree.cpio
     ```

   - para usar un dispositivo en lugar de un fichero, sustituir `tree.cpio` por `/dev/`*`dispositivo`*
2. *Copy-in*: extrae los ficheros de un archivo, con la opción `-i`

   ```
   $ cpio -idv < tree.cpio
   ```

   - la opción `d` crea los directorios al ir extrayendo
3. *Copy-pass*: usado para copiar ficheros de un árbol de directorios a otro, con la opción `-p`
   - Ejemplo: copia los ficheros del directorio actual y subdirectorios a un nuevo directorio `new-dir`

     ```
     $ find . -depth -print0 | cpio --null -pvd new-dir
     ```

   - la opción `-depth` procesa primero el contenido del directorio y después el directorio (mejor para restaurar)
   - las opciones `-print0` y `--null` evitan problemas con nombres de ficheros que contengan un carácter de *newline*
     - `-print0` termina los nombres de los ficheros con un '\0' en vez de '\n'
     - `--null` o `-0` lee una lista de ficheros terminados por un '\0'

- Para más opciones y uso de `cpio` ver la página de información: `info cpio`

Comando `afio`

Variación de `cpio`, con varias mejoras:

- Permite hacer copias multivolumen
- Permite archivar los ficheros comprimiendolos de uno en uno
  - No comprime los ficheros que no interesa comprimir por que ya lo están (reconoce por extensión)

- Permite verificar la copia con el original (opción `-r`)

Modos de funcionamiento similares a `cpio`

- `-o` guarda a archivo, `-i` extrae de un archivo y `-p` copia directorios
- otras opciones: `-r` verifica el archivo con el filesystem; `-t` muestra el contenido del archivo
- Ejemplos:
  - Salvar a disquete multivolumen comprimido

    ```
    $ find . | afio -ov -s 1440k -F -Z /dev/fd0
    ```

  - Comprobar con el original una copia comprimida en varios disquetes

    ```
    $ afio -rv -s 1440k -F -Z /dev/fd0
    ```

  - Muestra el contenido del archivo:

    ```
    $ afio -tv -s 1440k -F -Z /dev/fd0
    ```

  - Extrae el contenido del archivo

    ```
    $ afio -iv -s 1440k -F -Z /dev/fd0
    ```

  - Copia los ficheros del directorio actual y subdirectorios a un nuevo directorio `new-dir`

    ```
    $ find . -depth -print0 | afio -p0xa directorio_nuevo
    ```

- Para opciones ver la página de manual

Comando `dd`

Comando de copia y conversión de ficheros

- Sintaxis.

  `dd [if=`*`fichero_entrada`*`] [of=`*`fichero_salida`*`] [opciones]`

- Por defecto, copia de la entrada estándar a la salida estándar
- Algunas opciones:
  - `ibs=`*`b`* lee *`b`* bytes de cada vez (tamaño de bloque, por defecto 512)
  - `obs=`*`b`* escribe *`b`* bytes de cada vez

- `bs=`*b* lee y escribe *b* bytes de cada vez
- `cbs=`*b* especifica el tamaño del bloque de conversión
- `skip=`*n* salta *n* bloques del fichero de entrada antes de la copia
- `seek=`*n* salta *n* bloques del fichero de salida antes de la copia
- `count=`*n* copia sólo *n* bloques del fichero de entrada
- `conv=`*conversión* convierte el formato del fichero de entrada según el valor de *conversión*:
  - `ascii` Convierte EBCDIC a ASCII
  - `ebcdic` Convierte ASCII a EBCDIC
  - `swab` Intercambia cada par de bytes de la entrada
  - `lcase` Cambia las letras mayúsculas a minúsculas
  - `ucase` Cambia las letras minúsculas a mayúsculas
  - `noerror` Continúa después de producirse errores de lectura

- Ejemplo: imagen de floppy de 3.5, con 18 sectores por pista, dos cabezas y 80 cilindros:

  ```
  $ dd bs=2x80x18b if=/dev/fd0 of=/tmp/floppy.image
  ```

  - la `b` representan bloques de 512 bytes (en total 1474560 bytes)
  - la copia se realiza de una sola vez

- Ejemplo: extrae los datos de una cinta con error

  ```
  $ dd conv=noerror if=/dev/st0 of=/tmp/bad.tape.image
  ```

- Ejemplo: `tar` del directorio actual y copia en cinta en el sistema remoto

  ```
  $ tar cjf - . | ssh remoto dd of=/dev/st0
  ```

Comando `mt`

Permite la manipulación directa de la unidad de cinta

- Sintaxis.

  ```
  mt [-f unidad_de_cinta] operación [número]
  ```

- con `-f` indicamos la unidad de cinta a utilizar:
  - si se omite se toma la definida en la variable `TAPE`

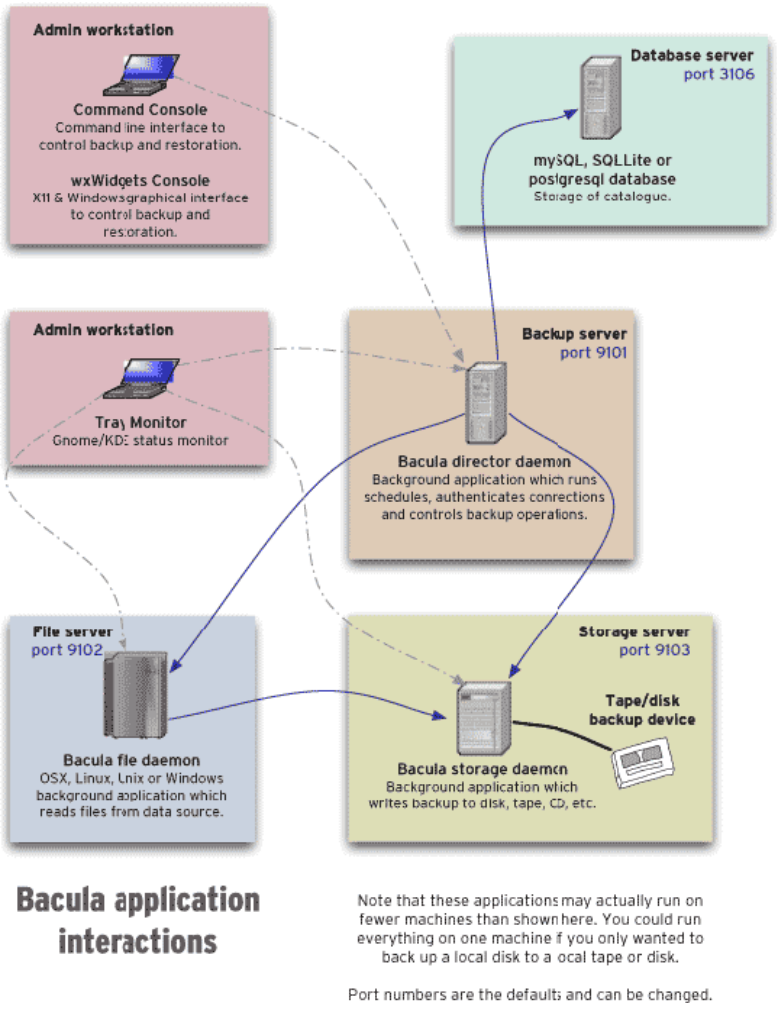
- Algunas operaciones:
  - `stat(us)` muestra el estado de la unidad de cinta
  - `rew(ind)` rebobina la cinta hasta el principio
  - `ret(ension)` alisa y da tensión a la cinta (rebobina hasta el principio, luego hasta el final y nuevamente al principio)
  - `erase` borra la cinta entera
  - `fsf`/`bsf` se avanza/retrocede el número de archivos especificado por *número*
  - `eom` salta hasta el final de parte grabada

6.3. Otras aplicaciones

Existen otros comandos que permiten hacer backups y sincronizar ficheros: `bacula`, `rdist`, `rsync`, etc.

Bacula

- Sofisticado sistema de backup en red con diseño modular
- Permite hacer copias de seguridad de todas las máquinas de una LAN a diferentes medios de backups (cinta, disco,...)
- Soporta MySQL, PostgreSQL o SQLlite para el catálogo
- Hace backups de sistemas UNIX, Linux y Windows
- Para más información, ver www.bacula.org/en/?page=documentation o la sección 9.8 del libro *UNIX System Administration Handbook*, Evi Nemeth et al.



Amanda

`Amanda`: *Advanced Maryland Automatic Network Disk Archiver*

- Sofisticado sistema de backup en red
- Permite hacer copias de seguridad de todas las máquinas de una LAN a una unidad de cinta en un servidor
- Está disponible en la mayoría de los UNIX y soporta muchos tipos de medios de backup
- Puede hacer uso de SAMBA para copias de sistemas Windows NT
- Se basa en `dump` y `tar`
- Para más información, ver www.amanda.org/docs

Flexbackup

Flexbackup Herramienta de backup flexible para instalaciones de pequeño y medio tamaño

- Más simple de configurar y utilizar que `Amanda` para sitios con un número no muy alto de sistemas
- Usa distintos formatos de archivo: `dump`, `afio`, `GNU tar`, `cpio`, `zip`, etc.
- Permite backups completos e incrementales, como `dump`
- Permite backups remotos a través de `rsh` o `ssh`
- Para más información, ver flexbackup.sourceforge.net

rdiff-backup

rdiff-backup copia un directorio en otro, permitiendo copias remotas

- Hace una copia exacta de los directorios (*mirror*), guardando las propiedades de los ficheros (propietario, permisos, etc.)
- Guarda las diferencias entre copias de los ficheros para poder recuperar un fichero antiguo (*incremental*)
- Sólo transmite las diferencias de los ficheros (similar a `rsync`)
- Para más información ver www.nongnu.org/rdiff-backup

DAR

DAR *Disk ARchiver* comando para hacer backups de árboles de directorios y ficheros

- Permite copiar un filesystem entero a un archivo
- Permite hacer backups completos y diferenciales
- Permite hacer copias multivolumen:
  - divide en archivo en varios ficheros (*slices*) parando antes de crear cada nuevo *slice*
  - interesante para hacer copias en floppy, CD o DVD
- Más información en: dar.linux.free.fr

Comando `rdist`

Permite distribuir ficheros desde un servidor central a varias máquinas

- sólo copia los ficheros modificados, preservando el propietario, grupo, modo y fechas de modificación
- las versiones actuales pueden funcionar sobre `ssh` (las antiguas funcionaban sobre `rlogin`, con problemas de seguridad)
- utiliza un fichero `distfile` que especifica las acciones a realizar
- Ejemplo de `distfile`

```
SYS_FILES = (/etc/passwd /etc/group /etc/mail/aliases)
HOME_DIRS = (/home/tomas /home/al*)
GET_ALL = (maquina1 maquina5 maquina6)
GET_SOME = (maquina2 maquina8)

all: ${SYS_FILES} -> ${GET_ALL}
notify tomas@localhost;
special /etc/mail/aliases ¨/usr/bin/newaliases¨;

some: ${SYS_FILES} -> ${GET_SOME}
except /etc/mail/aliases;
```

- ejemplo de uso:

```
# rdist -f distfile
```

Comando `rsync`

Similar a `rdist` aunque funciona de forma diferente

- más eficiente que `rdist`, sólo transmite las diferencias entre ficheros
- no usa fichero de configuración: funciona de forma similar a `rcp`
- ejemplo:

```
# rsync -av /home/tomas maquina1:/tmp
```

- ver la página de manual de `rsync` para más detalles

Unison

Aplicación para sincronizar ficheros y directorios entre sistemas

- puede sincronizar entre sistemas Windows y UNIX
- no requiere permisos de root
- permite sincronización en los dos sentidos
- las transferencias se optimizan usando una versión de `rsync`
- tiene un interfaz gráfico sencillo
- para ver un tutorial de uso, hacer:

```
$ unison -doc tutorial
```

Imágenes del sistema

Herramientas que nos permiten obtener imágenes completas del sistema para copias de seguridad o réplicas (clones)

Norton Ghost herramienta comercial de Symantec para copias de seguridad e imágenes del sistema

Partimage salva particiones completas a un fichero de imagen

- permite recuperar la partición completa en caso de errores
- permite realizar clones de un PC

Clonezilla aplicación opensource para hacer clones masivos

- permite hacer clones de múltiples PCs (40 o más) simultáneamente
- puede usar multicast para distribuir las imágenes
- basado en DRBL (*Diskless Remote Boot in Linux*) y Partclone

SystemImager herramienta para automatizar la instalación de Linux y la distribución de software en una red de PCs

- usado en clusters, granjas de servidores o redes en general

Tema 4: Servicios básicos de servidor a cliente

Administración de Sistemas e Redes

Tomás Fernández Pena

tf.pena@usc.es

Índice

1. Introducción

Dos tipos de servicios:

1. Servicios de Internet:
   - Servicios de ejecución remota: telnet, ssh
   - Servicios de transferencia de ficheros: ftp, sftp
   - Servicio de DNS
   - Servicio de Proxy
   - Servicio de correo electrónico: SMTP, POP, ...
   - Servicio Web
2. Servicios de intranet
   - Sistemas de archivos de red (NFS)
   - Servicio de información de red (NIS)
   - Servicio de directorio (LDAP)
   - Compartición Windows/Linux (Samba)

Los servicios de DNS, Web, Proxy y e-mail se tratan en la asignatura Administración Avanzada de Sistemas e Redes

2. Acceso remoto y transferencia de ficheros

Permiten acceder a un sistema remoto y transferir ficheros de/hacia este sistema

- Aplicaciones clásicas
  1. `telnet` (*TELetype NETwork*) permite conectarnos a otros ordenadores de la red como terminal remoto

2. `ftp` (*File Transfer Protocol* permite intercambiar ficheros entre distintos ordenadores de la red

- Problema: la información se transfiere en claro
- El uso de `telnet` y `ftp` se desaconseja
- Reemplazarlos por `ssh`, `scp`, `sftp`

1. `ssh` (*Secure Shell*) permite conectarnos a otro sistema encriptando toda la información
2. `scp`, `sftp` permiten la transferencia de ficheros de forma encriptada
   - `scp` similar a `cp` y `sftp` similar a `ftp`

2.1. Servicio de telnet y ftp

Los servicios TCP (telnet, ftp, talk, finger, etc.) son normalmente lanzados por el superdemonio de red `inetd` (o `xinetd`)

- El fichero de configuración es el `/etc/inetd.conf`
- Ejemplo de línea

```
telnet    stream  tcp  nowait  telnetd  /usr/sbin/in.telnetd
```

  - cuando `inetd` reciba una petición al puerto `telnet` abre un socket tipo *stream* y ejecuta `fork()` y `exec()` del programa `/usr/sbin/in.telnetd`, bajo la identidad del usuario `telnetd`
  - `nowait` indica que el servidor puede continuar procesando conexiones en el socket

- Versión mejorada de `inetd`: `xinetd`
- Para mayor control usar *TCP Wrapper* (programa `tcpd`)
  - Permite conceder/denegar acceso a determinados hosts/redes mediante los fichero `/etc/hosts.allow` y `/etc/hosts.deny`

Servicio de telnet

Instalación de un servidor telnet

- Descargar el paquete `telnetd`
  - El paquete actualiza el `/etc/inetd.conf`
  - Por defecto usa TCP wrappers
  - El servidor escucha el puerto 23

  Desistalar el servicio telnet

  - Desinstalar el paquete `telnetd`, o
  - Comentar la línea correspondiente en `/etc/inetd.conf`

Servicio de FTP

Transfiere ficheros a/desde un host remoto

- Permite usuarios registrados o anónimos (*anonymous*)
- Utiliza dos puertos: 21 (conexión de control) y 20 (conexión de datos)
- Dos modos de funcionamiento:
  1. Activo (modo por defecto en el comando ftp)
     - El servidor inicia la conexión de datos desde su puerto 20 a un puerto > 1023 del cliente
     - Problema con los firewalls en el cliente
  2. Pasivo (modo recomendable, por defecto en navegadores)
     - El cliente inicia las conexiones de control y datos
     - No se utiliza el puerto 20
     - No tiene problema con los firewall

Instalación de un servidor ftp básico

1. Instalar el paquete `ftpd`
   - El paquete actualiza el `/etc/inetd.conf`
   - Por defecto usa TCP wrappers
   - Podemos denegar el acceso ftp a ciertos usuarios incluyéndolos en el fichero `/etc/ftpusers`

Servicio de FTP avanzado

Servidores avanzados de FTP

- Proporcionan numerosas facilidades, tanto para ftp normal cono anónimo
- Existen numerosos servidores comerciales u open source: Wu-FTPD, Pure-FTPd, ProFTPD, wzdftpd, vsftpd
- Estos servidores proporcionan normalmente:
  - Operación a través de inetd o *standalone*
  - Servidores FTP virtuales (varios servidores de FTP anónimos en el mismo host)
  - Usuarios FTP virtuales (cuentas ftp diferentes de las cuentas del sistema)
  - Facilidades para registro y monitorización de accesos
  - Facilidades para controlar y limitar accesos
  - Comunicación encriptada

2.2. SSH

SSH: Shell seguro

- Permite comunicarnos de forma segura con un servidor remoto
  - Permite abrir sesiones o transferir ficheros (`scp` o `sftp`)
  - Reemplazo de `rlogin`, `telnet` o `ftp`
  - Todos los datos viajan encriptados
  - Dos versiones SSH-1 y SSH-2:
    - Recomendable SSH-2
    - Versión open-source OpenSSH
- Paquetes Debian:
  - Ciente: `openssh-client`
  - Servidor: `openssh-server`

Modos de autenticación mediante SSH

SSH soporta 4 modos de autenticación:

1. Si el nombre del host remoto desde el cual un usuario se conecta al servidor esta listado en `~/.rhosts`, `~/.shosts`, `/etc/hosts.equiv` o `/etc/shosts.equiv` el usuario remoto puede entrar sin contraseña
   - Método absolutamente desaconsejado
2. Igual que el anterior pero la clave pública del host remoto debe aparecer en `/etc/ssh_known_hosts` o `~/.ssh/known_hosts`
   - No demasiado seguro (si el host remoto se ve comprometido, el servidor local queda comprometido)
3. La clave pública del usuario remoto debe estar en `~/.ssh/authorized_keys`
   - El usuario remoto debe tener acceso a su clave privada
   - Método más seguro, pero un poco incomodo
4. Acceso mediante contraseña (modo por defecto)
   - Menos seguro que el anterior

Opciones para autenticación

Fichero de configuración del servidor ssh: `/etc/ssh/sshd_config`

Opción	**M**	**Dfto.**	**Significado**
RhostsRSAAuthentication	2	`no`	Si `yes` permite autenticación por host (SSH-1)
HostbasedAuthentication	2	`no`	Si `yes` permite autenticación por host (SSH-2)
IgnoreRhosts	2	`yes`	No usa los ficheros `~/.rhosts` y `~/.shosts`
IgnoreUserKnownHosts	2	`no`	Ignora el fichero `~/.ssh/known_hosts`
RSAAuthentication	3	`yes`	Autenticación de clave pública de usuario (SSH-1)
PubkeyAuthentication	3	`yes`	Autenticación de clave pública de usuario (SSH-2)
PasswordAuthentication	4	`yes`	Autenticación mediante contraseña
UsePAM	2,3,4	`no`	Usa PAM para autenticación

Otras opciones de configuración del servidor

Otras opciones en `/etc/ssh/sshd_config`

Opción	Dfto.	Significado
Port	22	Puerto (puede ser interesante cambiarlo a >1024)
Protocol	2,1	Protocolo aceptado (más seguro sólo 2)
ListenAddress	Todas	Dirección local por la que escucha
PermitRootLogin	`yes`	Permite acceder al root
X11Forwarding	`no`	Permite forwarding X11

Para más opciones `man sshd_config`

Opciones para el cliente

Fichero de configuración del cliente ssh: `/etc/ssh/ssh_config` o `~/.ssh/config`

- En este fichero se especifican opciones para los comandos `ssh`, `scp` o `sftp`
- Algunas de estas opciones se pueden especificar en el momento de ejecutar el comando, p.e.

```
$ ssh -p port servidor # Indica otro puerto
```

Algunas opciones:

Opción	Dfto.	Significado
Hosts		Host para los que se aplican las opciones (* implica todos)
Port	22	Puerto por defecto
Protocol	2,1	Protocolo usado por defecto
Cipher[s]		Mecanismos de cifrado usados
ForwardX11	`no`	Reenvío X11

Para más opciones `man ssh_config` y `man ssh`

Otros comandos

- `ssh-keygen` generación y gestión de claves públicas/privadas para SSH
  - Permite claves RSA o DSA (DSA sólo SSH-2, por defecto RSA-2)
  - Ficheros (para SSH-2)

    - Clave privada: `~/.ssh/id_rsa` o `~/.ssh/id_dsa` (`~/.ssh/identity` para SSH-1)
    - Clave pública: `~/.ssh/id_rsa.pub` o `~/.ssh/id_dsa.pub` (`~/.ssh/identity.pub` para SSH-1)
  - La clave privada debe tener una *passphrase* de longitud arbitraria
    - Puede cambiarse con la opción `-p`

- `ssh-agent` Agente de autenticación
  - Mantiene en memoria la clave privada
  - Evita tener que escribir la *passphrase* cada vez que usemos ssh
  - Habitualmente, si entramos en X11 se activa automáticamente
    - Opción `use-ssh-agent` de `/etc/X11/Xsession.options` (ver `man xsession.options`)
  - Para activarlo en consola usar (como usuario)

    ```
    eval $(ssh-agent)
    ```

  - Define las variables `SSH_AUTH_SOCK` y `SSH_AGENT_PID`

- `ssh-add` Añade las claves privadas al agente
  - Uso:

    ```
    ssh-add [opciones] [-t life] [fichero]
    ```

  - Por defecto añade los ficheros `~/.ssh/id_rsa`, `~/.ssh/id_dsa` y `~/.ssh/identity`, pidiendo las correspondientes *passphrases*
  - Pueden añadirse múltiples claves
    - En una conexión se prueban las diferentes claves hasta que coincide
  - Algunas opciones:
    - `-l` Muestra las identidades añadidas
    - `-t life` Especifica un tiempo de vida de la identidad

3. Sistemas de archivos de red (NFS)

NFS (*Network File System*) permite compartir sistemas de ficheros en la red

- Introducido por Sun MicroSystems en 1985, y soportado por todos los Unixes

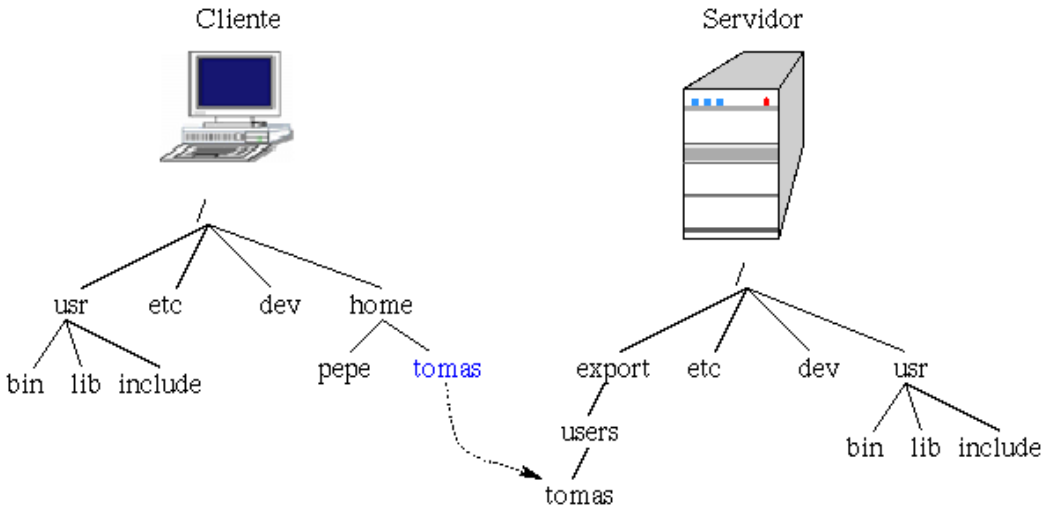
- Versiones principales: NFSv2 y NFSv3
  - NFSv4 más reciente (RFC 3530, incluido en kernel 2.6, última revisión 4.1)
- NFSv2 y 3: protocolo sin estado: no hay pérdida de información si el servidor cae
  - NFSv4 incorpora estado: mecanismo complejo de recuperación de caídas
- Comunicación mediante TCP o UDP
  - normalmente UDP (NFSv4 usa TCP)
- Dos tipos de servidores en Linux:
  - servidor en espacio de usuario: más lento y con problemas
  - servidor en modo kernel: más rápido, menos características (versión por defecto)

Para más información:

- capítulo 16 (*The Network File System*) del libro "Linux System Administration Handbook" (2a ed.), Evi Nemeth et.al.
- Linux NFS-HOWTO
- nfs.sourceforge.net

3.1. Características principales

Ejemplo de funcionamiento



Procesos implicados

NFS se basa en RPC (*Remote Procedure Call*)

- el servicio *portmap* (también llamado *rpcbind*) debe estar disponible y activo
  - convierte números de programas RPC en números de puertos
  - utiliza el puerto 111
  - necesario para aplicaciones que usen RPC
  - el comando `rpcinfo` nos muestra información RPC
  - en Debian, paquete `portmap`

Otros demonios necesarios:

- `rpc.nfsd`: implementa la parte de usuario del servidor NFS (atender y resolver las peticiones de acceso del cliente a archivos situados en el directorio remoto)
- `rpc.mountd`: proceso que recibe la petición de montaje desde un cliente NFS y chequea para mirar si coincide con un sistema de ficheros actualmente exportado, y si el cliente tiene permisos suficientes para montar dicho directorio
- `rpc.rquotad`: proporciona información de cuotas a usuarios remotos
- `rpc.statd`: implementa el protocolo NSM (*Network Status Monitor*); proporciona un servicio de notificación de reinicio, cuando NFS cae; lo usa el servicio de bloqueo de ficheros *lockd*

- `rpc.lockd`: servicio de bloqueo de ficheros (*NFS lock manager*, NLM); no necesario en kernels modernos (>= 2.4) en los que el boqueo es realizado por el kernel

NFSv4 no usa `portmap`, ni los demonios `rpc.mountd` y `rpc.statd`

- Usa autenticación basada en Kerberos mediante los siguientes servicios:
  - `rpcsec_gss` (cliente `rpc.gssd`, servidor `rpc.svcgssd`): autenticación de la conexión cliente-servidor
  - `rpc.idmapd`: mapeo entre UIDs (o GIDs) y nombres de usuario (o nombres de grupos)

3.2. Instalación de NFS en Debian

Veremos como instalar un servidor y con cliente NFS v3 y v4 en Debian

- Los paquetes a instalar para las dos versiones son los mismos: `nfs-kernel-server` y `nfs-common` (este último suele estar instalado por defecto)
  - `nfs-kernel-server` proporciona `rpc.nfsd`, `rpc.mountd`, y para NFSv4 `rpc.svcgssd`
  - `nfs-common` proporciona `rpc.lockd`, `rpc.statd`, y para NFSv4 `rpc.gssd` y `rpc.idmapd`
- El fichero básico de configuración es el mismo en las dos versiones: `/etc/exports`

Servidor NFSv3

1. Configurar los directorios a exportar: fichero `/etc/exports`
   - Ejemplo de fichero `/etc/exports`

```
/projects       (ro) proj*.usc.es(rw,no_subtree_check)
/home           193.144.84.0/24(rw,no_subtree_check,root_squash,sync
/pub            (ro,all_squash)
```

   - exporta `/projects` de sólo lectura para todo el mundo y lectura/escritura para los sistemas `proj*.usc.es`
   - Algunas opciones de la exportación:
     - `rw/ro` exporta el directorio en modo lectura/escritura o sólo lectura

- `root_squash` mapea los requerimientos del UID/GID 0 al usuario *anónimo* (por defecto usuario *nobody*, con UID/GID 65534); es la opción por defecto
- `no_root_squash` no mapea root al usuario anónimo
- `all_squash` mapea todos los usuarios al usuario anónimo
- `squash_uids/squash_gids` especifica una lista de UIDs o GIDs que se deberían trasladar al usuario anónimo

  ```
  squash_uids=0-15,20,25-50
  ```
- `anonuid/anongid` fija el UID/GID del usuario *anónimo* (por defecto 65534)
- `subtree_check/no_subtree_check` con `subtree_check`, si se exporta un subdirectorio (no un filesystem completo) el servidor comprueba que el fichero solicitado por el cliente esté en el subdirectorio exportado; con `no_subtree_check` (opción por defecto) se deshabilita ese chequeo
- `sync` modo síncrono: requiere que todas las escrituras se completen antes de continuar; es opción por defecto
- `async` modo asíncrono: no requiere que todas las escrituras se completen; más rápido, pero puede provocar pérdida de datos en una caída
- `secure` los requerimientos deben provenir de un puerto por debajo de 1024
- `insecure` los requerimientos pueden provenir de cualquier puerto

- Para más opciones `man exports`
- Cada vez que se modifica este fichero se debe ejecutar el comando `exportfs` para actualizar el servidor

  ```
  # exportfs -ra
  ```

  - ver `man exportfs` para opciones del comando

2. Iniciar el demonio:

   ```
   # service nfs-kernel-server start
   ```

3. Comprobar los directorios exportados con `showmount`

   ```
   # showmount --exports localhost
   ```

- `showmount` muestra información de un servidor NFS: directorios que exporta, directorios montados por algún cliente y clientes que montan los directorios

4. Podemos ver las estadísticas del servidor NFS con `nfsstat`

Servidor NFSv4

1. Los exports de NFSv4 deben residir en un pseudodirectorio, donde los directorios reales a exportar se montan con la opción `--bind`, por ejemplo para exportar `/home`

```
# mkdir /export
# mkdir /export/home
# mount --bind /home /export/home
```

2. La opción `bind` permite remontar un directorio en otro sitio

   - Para que este montado permanezca, añadir al `fstab` la siguiente línea:

```
/home    /export/home    none    bind    0    0
```

3. Fichero `/etc/exports` en NFSv4

```
/export          193.144.84.0/24(rw,fsid=0,crossmnt,no_subtree_check,sync)
/export/home     193.144.84.0/24(rw,no_subtree_check,root_squash,sync)
```

4. Nuevas opciones de la exportación:

   - `fsid=0` designa este path como la raíz de los directorios exportados por NFSv4
   - `crossmnt` permite que los directorios debajo del raíz se muestren adecuadamente (alternativamente, se puede poner la opción `nohide` en cada uno de esos directorios)

Cliente NFS

El cliente NFS en Linux está integrado en el nivel del Sistema de Ficheros Virtual (VFS) del kernel

- no necesita un demonio particular de gestión (en otros UNIX, demonio *biod*)

Instalación:

1. Instalar (si no está ya instalado) el paquete `nfs-common`

2. Montar los directorios remotos con `mount -t nfs` (para v3) o `mount -t nfs4` (para v4), o añadir una entrada en `fstab` (ver Tema 3: Montado de los sistemas de ficheros). NOTA: en versiones actuales de Linux la opción `-t nfs` intenta montar con NFSv4 y si falla, pasa a NFSv3.

   - Ejemplo de uso con `mount` (IP servidor NFS 193.144.84.1):

     ```
     # mount -t nfs4 193.144.84.1:/home /mnt/home
     ```

   - Ejemplo de entrada en `fstab`

     ```
     193.144.84.1:/home      /home     nfs4     rw,auto     0 0
     ```

   - Automount se usa frecuentemente con NFS (ver la parte de Autofs en Tema 3: Montado de los sistemas de ficheros: Autofs)

Opciones particulares de montado con NFS:

- `rsize=`*n*`/wsize=`*n* especifican el tamaño del datagrama utilizado por los clientes NFS cuando realizan peticiones de lectura/escritura (pueden ajustarse para optimizar)

- `hard` el programa accediendo al sistema de ficheros remoto se colgará cuando el servidor falle; cuando el servidor esté disponible, el programa continuará como si nada (opción más recomendable)

- `soft` cuando una petición no tiene respuesta del servidor en un tiempo fijado por `timeo=`*t* el cliente devuelve un código de error al proceso que realizó la petición (puede dar problemas)

Para ver más opciones, ver `nfs(5)`

3.3. Consideraciones de seguridad en NFS

NFS no fue diseñado pensando en la seguridad:

- Los datos se transmiten en claro

- Usa el UID/GID del usuario en el cliente para gestionar los permisos en el servidor:

  - El usuario con UID n en el cliente obtiene permisos de acceso a los recursos del usuario con UID n en el servidor (aunque sean usuarios distintos)

    - Un usuario con acceso a root en un cliente podría acceder a los ficheros de cualquier usuario en el servidor (no a los de root, si se usa la opción `root_squash`)

Precauciones básicas:

1. Usar NFS sólo en Intranets seguras, donde los usuarios no tengan acceso de administrador en sus sistemas
2. Evitar el acceso a NFS desde fuera de la Intranet
    - Bloquear los puertos TCP/UDP 111 (*portmap*) y 2049 (*nfs*)
3. Usar NFSv4 con Kerberos, que incluye autenticación de hosts y cifrado
4. Usar versiones seguras de NFS (Secure NFS) u otros sistemas de ficheros (p.e. Self-certifying File System, SFS)

Ver NFS howto y http://www.cert.org/tech_tips/unix_configuration_guidelines.html#A13

4. Servicios de directorio

Necesidad de mantener una configuración única a través de múltiples sistemas

- Compartición de los ficheros de configuración
- Ficheros a compartir:
    - `/etc/passwd`, `/etc/shadow`, `/etc/group`, etc.
- Mecanismos de compartición:
    - Copia de ficheros de un servidor central al resto de los equipos mediante `rdist` o `rsync`
    - Utilización de un servidor de dominio, que centralice esa información
        - NIS: Network Information Service
        - LDAP: Lightweight Directory Access Protocol

Concepto de dominio

Dominio conjunto de equipos interconectados que comparten información administrativa (usuarios, grupos, contraseñas, etc.) centralizada

- Necesidad de uno (o varios) servidores que almacenen físicamente dicha información y que la comunique al resto cuando sea necesario
- Normalmente se usa un esquema cliente/servidor
  - p.e. un usuario se conecta en un sistema cliente y este valida las credenciales del usuario en el servidor
- En Windows 2000, la implementación del concepto de dominio se realiza mediante el denominado Directorio Activo (*Active directory*)
  - Basado en LDAP y DNS
- En UNIX, el servicio clásico de gestión de dominios es NIS
  - NIS se considera bastante obsoleto
- Existen implementaciones libre del protocolo LDAP para Unix (`openLDAP`)
  - más potente y escalable que NIS para la implementación de dominios

4.1. Servicio de Información de Red NIS

Desarrollado por Sun Microsystem en los años 80

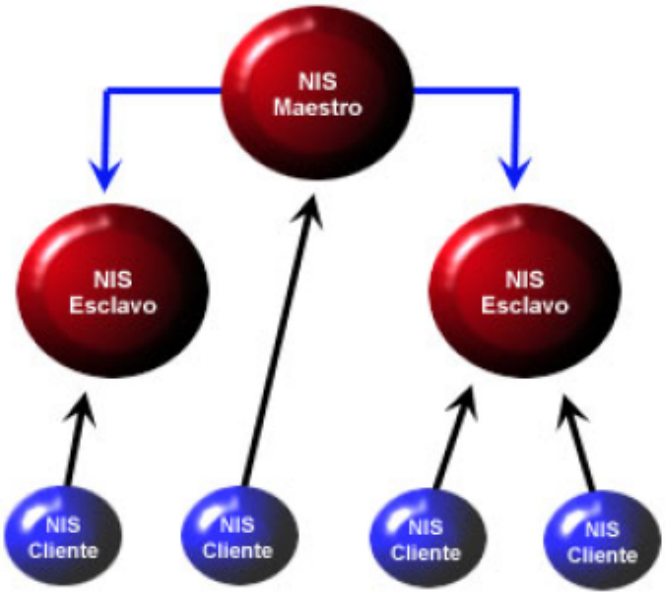
- Nombre original *Yellow Pages* (modificado por razones legales)
- Muy popular como sistema de administración de dominios en UNIX
- A principios de los años 90, versión NIS+ para Solaris 2.x
  - muy diferente de NIS
  - incorpora soporte para encriptación y autentificación de datos
  - complejo y poco soportado
- Para más información:
  - The Linux NIS(YP)/NYS/NIS+ HOWTO
  - Debian NIS HOWTO
  - Introducción a NIS y NFS

Funcionamiento básico de NIS

Base de datos distribuida

- Un servidor (*master*) mantiene los ficheros de configuración de los sistemas (`/etc/passwd`, `/etc/group`, etc.)
  - cada archivo de configuración se convierte en una o más tablas (*mapas* NIS) de una base de datos
  - esos mapas se guardan en un formato binario llamado DBM (*DataBase Management*)
  - el servidor NIS maestro debería tener ambas, las tablas ASCII y las DBM
  - en una red debe haber al menos una máquina actuando como un servidor NIS maestro
- Los clientes hablan directamente con el servidor NIS para leer la información almacenada en sus bases de datos DBM
- Pueden existir servidores NIS esclavos:
  - tienen copias de las bases de datos NIS
  - reciben estas copias del servidor NIS maestro cada vez que se realizan cambios a las bases de datos maestras
- Un servidor maestro y sus servidores esclavos y clientes constituyen un dominio NIS
  - una red puede tener múltiples servidores NIS, cada uno sirviendo a un dominio NIS diferente

Esquema de un dominio NIS



Comandos básicos de NIS

NIS incluye un conjunto amplio de comandos y demonios, algunos de los cuales son:

- `ypserv` demonio de servidor
- `ypbind` demonio de cliente
- `domainname` establece el nombre del dominio
- `ypinit` configura un servidor como maestro o esclavo
- `ypxfr` descarga un mapa desde servidor maestro (en los esclavos)
- `yppush` ejecutado en el maestro, hace que los esclavos actualicen sus mapas
- `ypwhich` muestra el sombre del servidor NIS
- `ypcat` muestra las entradas de un mapa
- `yppasswd` cambia la contraseña en la base de datos de NIS
- `ypchfn` cambia el campo GECOS en la base de datos de NIS
- `ypchsh` cambia el login shell en la base de datos de NIS

Instalación de NIS en Debian

El proceso de puesta en marcha de NIS depende de la distribución

- veremos como instalar un servidor maestro y un cliente en Debian

Servidor maestro

1. Instalar el paquete `nis`
   a) Indicar un nombre de dominio NIS (que no tiene que corresponder con el dominio de RED)
      - El nombre de dominio puede cambiarse con `domainname`
   b) La configuración puede tardar, ya que intenta iniciarse como cliente NIS y se queda buscando un servidor
2. Cambiar el fichero `/etc/default/nis`
   - debemos poner `NISSERVER=master`
3. En el fichero `/etc/ypserv.securenets` añadir el número de la red local, para permitir acceso exclusivo a los sistemas de esa red

   ```
   255.255.255.0        192.168.0.0
   ```

4. Ejecutar `/usr/sbin/ypserv` para iniciar el servidor
5. Editar (si es necesario) el fichero `/var/yp/Makefile`
   - permite configurar características generales así como las tablas a partir de las cuales se crean los mapas NIS
   - haciendo `make` en ese directorio se crean los mapas que se guardan en `/var/yp/`*`dominio`*
   - cada vez que se modifique alguna tabla (p.e. añadiendo un nuevo usuario), debemos hacer `make` para actualizar los mapas NIS
6. Ejecutar `/usr/lib/yp/ypinit -m` para que el sistema se configure como servidor maestro
   - no añadir ningún servidor NIS más (`Ctrl-D`)
7. Podemos comprobar que funciona bien haciendo un `ypcat` de alguno de los mapas (p.e. `ypcat passwd`)
8. Por último, el servidor debe configurarse también como cliente
   - seguir los pasos de la siguiente sección

Cliente NIS

1. Eliminar de los ficheros locales los usuarios, grupos y otra información que queramos que sea accesible por NIS (sólo en los clientes)
2. Instalar el paquete `nis` e indicar el nombre del dominio NIS
3. Si se desea, cambiar `/etc/yp.conf` para especificar el servidor NIS concreto
   - por defecto, busca el servidor mediante un broadcast
4. Modificar el archivo `/etc/nsswitch.conf` para que busque `passwd`, `group` y `shadow` por NIS:

   ```
   passwd:       files nis
   group:        files nis
   shadow:       files nis
   ```

   - El formato y opciones de ese fichero lo vimos en el Tema 5: Ficheros de configuración de red
   - Alternativamente, se puede dejar el modo `compat` añadiendo al final de los ficheros `passwd`, `shadow` y `group` del cliente un `+`, para indicar que vamos a usar NIS
   - este método permite incluir/excluir determinados usuarios
   - ver NIS-HOWTO: Setting up a NIS Client using Traditional NIS

Fichero `/etc/netgroup`

NIS introduce el concepto de *netgroups*

- grupos de usuarios, máquinas y redes que pueden ser referenciadas como un conjunto
- se definen en el fichero `/etc/netgroup`, en principio, sólo en el servidor NIS maestro

Formato de una entrada en `netgroup`

```
netgroup_name (host, user, NIS_domain), ...
```

- `host` nombre de una máquina en el grupo
- `user` nombre de login de un usuario de la máquina `host`

- `NIS_domain` dominio NIS nombre del dominio NIS

Pueden dejarse entradas en blanco o con un guión:

- Una entrada en blanco implica cualquier valor, p.e.
  - `(doc19, , )` indica todos los usuarios del host `doc19`
- Una entrada con un guión (-) implica campo sin valor, p.e.
  - `(-, pepe, )` indica el usuario `pepe` y nada más

Ejemplo de un fichero `/etc/netgroup`

```
sysadmins    (-,pepe,) (-,heidis,) (-,jnguyen,) (-,mpham,)
servers      (numark,-,) (vestax,-,)
clients      (denon,-,) (technics,-,) (mtx,-,)
research     (-,boson,) (-,jyom,) (-,weals,) (-,jaffe,)
allusers      sysadmins research
allhosts      servers clients
```

Estos *netgroups* pueden usarse en varios ficheros del sistema para definir permisos:

- con NIS en modo `compat`, p.e. añadiendo `+@sysadminds` en `/etc/passwd` daríamos permiso de acceso a los usuarios definidos como `sysadmins`
- en el fichero `/etc/exports`, para indicar grupos de máquinas a las que exportar un directorio por NFS

  ```
  /home        allhosts(rw,root_squash,sync)
  ```

- en los ficheros `/etc/hosts.allow` y `/etc/hosts.deny` de los TCP Wrappers
- etc.

5. Servicio de directorio: LDAP

LDAP: Protocolo Ligero de Acceso a Directorios (*Lightweight Directory Access Protocol*)

- Protocolo de red para consulta y modificación de datos de directorios X.500

    - X.500: Estándares de la ITU-T para servicios de directorio
    - Define, entre otros, un protocolo de acceso a directorios llamado DAP (*Directory Access Protocol*)
    - DAP definido sobre la pila completa de niveles OSI: costoso y complejo
- LDAP es una alternativa ligera al protocolo DAP
    - Opera directamente sobre TCP/IP
    - Actualmente, la mayoría de servidores de directorio X.500 incorporan LDAP como uno de sus protocolo de acceso
- Diferentes implementaciones del protocolo LDAP
    - Microsoft Server Active Directory
    - NetIQ eDirectory
    - Oracle Internet Directory
    - IBM Security Directory Server
    - Apache Directory Server
    - 389 Directory Server
    - Red Hat Directory Server
    - OpenLDAP
- Más información sobre LDAP
    1. LDAP Linux HOWTO
    2. IBM RedBooks: Understanding LDAP - Design and Implementation
    3. Recursos, ayudas, ...: ldapman.org

5.1. OpenLDAP

- Implementación *open source* del protocolo LDAP
- Basado en software desarrollado en la Universidad de Michigan
- Incluye cuatro componentes principales
    - slapd - demonio LDAP stand-alone (servidor)

    - slurpd - demonio de replicación y actualización de LDAP
    - librerías que implementan el protocolo LDAP
    - utilidades, herramientas, y clientes básicos
- Más información sobre la configuración de OpenLDAP en el OpenLDAP Administrator's Guide

5.2. Modelo de datos de LDAP

Un directorio es una base de dato optimizada para lectura, navegación y búsqueda

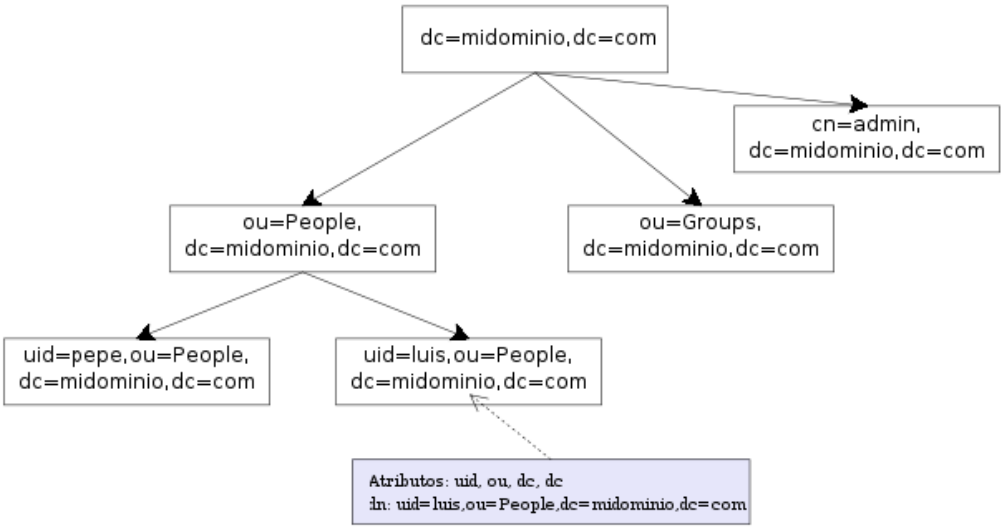
- la información se almacena de manera jerárquica
- generalmente no se soportan transacciones complejas ni sistemas de recuperación
- las actualizaciones son cambios simples
- proporcionan respuestas rápidas a grandes volúmenes de búsquedas
- el directorio puede estar replicado y/o distribuido entre varios sistemas (p.e. DNS)

LDAP organiza el directorio como una estructura jerárquica de entradas (nodos) en forma de árbol

- Cada entrada posee un conjunto de atributos, que pueden ser de diferentes tipos
    - cada atributo se identifica por su tipo y uno o más valores
    - los tipos son normalmente palabras nemotécnicas, como `uid` (identificador de usuario), `cn` (*common name*), `c` (*country*), `dc` (*domain component*), etc.
    - los diferentes atributos de un nodo están determinados por la clase a la que pertenece
    - las clases permiten definir entradas con diferente información: clases para personas, para equipos, administrativas, etc.
    - las clases se definen mediante ficheros de *esquema* (*schema*)
- Cada nodo debe poseer un nombre único: *nombre distinguido* o `dn` (*distinguished name*)

- el `dn` identifica de forma unívoca cada objeto en la base de datos

Ejemplo: árbol de usuarios y grupos en LDAP, basado en nombres de dominios de Internet:



- cada nodo puede tener varios atributos, p.e. el nodo *pepe* podría tener los siguientes atributos:

```
dn: cn=Jose Pena,ou=People,dc=midominio,dc=com
objectClass: account
uid: pepe
sn: Pena
description: alumno
mail: pepe@midominio.com
```

- el formato en el que se muestran los atributos del objeto se denomina LDIF (*LDAP Data Interchange Format*)
  - formato de intercambio de datos para importar y exportar datos a un servidor LDAP

5.3. Instalación de un servidor LDAP

Describiremos como montar un servidor LDAP simple que nos permita la gestión de usuarios y grupos

- el servidor mantendrá la lista de usuarios y grupos del dominio
- en los clientes la autenticación de usuarios y los permisos se basará en el servidor LDAP

Pasos para la instalación del servidor en Debian

1. Instalar los paquetes `slapd` (servidor OpenLDAP) y `ldap-utils` (utilidades del paquete OpenLDAP: `ldapsearch`, `ldapadd`)
   - En la configuración, elegir una contraseña para el administrador del LDAP (no tiene que ser la contraseña de root del sistema)
   - El comando `slapcat` permite ver la estructura creada
     - comprobar que la estructura creada por defecto es correcta
   - Podemos hacer búsquedas sencillas con `ldapsearch`

     ```
     # ldapsearch -x -b dc=midominio,dc=com '(cn=admin)'
     ```

2. Ficheros de configuración:

   a) Fichero de opciones del demonio `/etc/default/slapd`
   - Permite, entre otras cosas, especificar las interfaces donde se desea que escuche ldap (por defecto, todas las interfaces usando TCP puerto 389, `URI ldap://389`)
   - La variable `SLAPD_SERVICES` indica los mecanismos de escucha de slapd
     - Puede aceptar conexiones estándar (`ldap://`), conexiones seguras con SASL (*Simple Authentication and Security Layer*), usando `ldaps:///` o peticiones realizadas desde sockets UNIX (`ldapi:///`)
   - Indicar que acepte conexiones estándar en la red del cliente (`ldap://172.23.20.1:389/`), y reiniciar el servicio (`service slapd restart`)

   b) Fichero de configuración del servidor `/etc/ldap/sldap.conf`
   - Fichero de configuración del demonio `slapd` (en principio, no es necesario cambiarlo)
   - En versiones recientes, cambiado por ficheros LDIF en el directorio `/etc/ldap/slapd.d`
   - Para más info, ver `man slapd.conf`

c) Fichero `/etc/ldap/ldap.conf`

- Fichero de configuración global para los clientes LDAP
- Permite especificar la base por defecto, el servidor LDAP, etc. (cambiarlo para poner nuestra configuración, comprobar que tiene permisos 644)
- Para más info, ver `man ldap.conf`

3. Crear la estructura de la base de datos: crearemos los nodos de `People` y `Group` del árbol LDAP

   a) Crear el siguiente fichero `estructura.ldif` en formato LDIF:

```
dn: ou=People,dc=midominio,dc=com
objectClass: top
objectClass: organizationalUnit
ou: People

dn: ou=Group,dc=midominio,dc=com
objectClass: top
objectClass: organizationalUnit
ou: Group
```

   b) Añadir los nodos a la base de datos:

```
# ldapadd -x -D 'cn=admin,dc=midominio,dc=com' -W
-f estructura.ldif
```

   - `-x` autenticación simple sin SASL
   - `-D` nombre distinguido con el que nos conectamos a LDAP (ponemos el del administrador)
   - `-W` pide la contraseña de forma interaciva
   - `-f` fichero a cargar

4. Añadir un usuario y un grupo a la base de datos

   a) Crear un fichero como este, que tiene la información para un usuario y un grupo:

```
dn: cn=Jose Pena,ou=People,dc=midominio,dc=com
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
uid: pepe
cn: Jose Pena
```

```
uidNumber: 2000
gidNumber: 2000
homeDirectory: /home/pepe
loginShell: /bin/bash
gecos: Jose Pena, Despacho 22,,

dn: cn=pepe,ou=Group,dc=midominio,dc=com
objectClass: top
objectClass: posixGroup
cn: pepe
gidNumber: 2000
```

- define un usuario `pepe` y un grupo `pepe` con información similar a la aparece en el fichero `/etc/passwd`
- puede añadirse más información, como la que aparece en el fichero `/etc/shadow`: campos `shadowMax`, `shadowExpire`, `shadowWarning`, etc.

b) Añadir el fichero con:

```
# ldapadd -x -D 'cn=admin,dc=midominio,dc=com' -W
-f user-group.ldif
```

c) Probarlo haciendo búsquedas, tipo:

```
# ldapsearch -x uid=pepe
```

d) Añadir una contraseña al usuario: puede hacerse directamente metiendo un campo `userPassword` en el fichero anterior, pero es preferible hacerlo mediante el comando `ldappasswd`

```
# ldappasswd -x 'cn=Jose Pena,ou=People,dc=midominio,dc=com'
-D 'cn=admin,dc=midominio,dc=com' -W -S
```

5. Por último, también podemos querer instalar el servidor como cliente, por lo que debemos seguir los pasos indicados en la sección de instalación de un cliente

5.4. Migración desde ficheros o NIS

Si tenemos un sistema configurado mediante ficheros (`passwd`, `shadow`, etc.) o NIS, migrar a LDAP puede ser laborioso

- los scripts del paquete *MigrationTools* son de gran ayuda

Migración desde ficheros

Pasos para migrar los usuarios definidos en `/etc/passwd` a LDAP

1. Instalar el paquete `migrationtools`
2. Modificar el fichero `/usr/share/migrationtools/migrate_common.ph` para indicar el dominio y la base LDAP

   ```
   $DEFAULT_MAIL_DOMAIN = "midominio.com"
   $DEFAULT_BASE = "dc=midominio,dc=com"
   ```

3. Descomentad y poned los valores adecuados en las variables `$IGNORE_UID_...` y `$IGNORE_GID_...` para que no considere los usuarios y grupos del sistema
4. Utilizar los diferentes scripts del directorio `/usr/share/migrationtools` para incorporar la información del sistema al directorio
   - `migrate_base.pl` genera entradas correspondientes a las unidades organizativas por defecto: `People`, `Group`, `Hosts`, etc. (ya lo tenemos)
   - Otros scripts generan entradas asociadas a diferentes ficheros del sistema

Script	**Migra**
migrate_passwd.pl	/etc/passwd y /etc/shadow
migrate_group.pl	/etc/group
migrate_hosts.pl	/etc/hosts
migrate_fstab.pl	/etc/fstab
...	

   - Estos scripts generan ficheros LDIF que podemos añadir a la base LDAP con `ldapadd`
   - Ejemplo: migración de los usuarios
     - crear el fichero ldif con los usuarios con el comando

       ```
       # ./migrate_passwd.pl /etc/passwd ./passwd.ldif
       ```

     - añadir el fichero generado a la base LDAP con

       ```
       # ldapadd -x -D 'cn=admin,dc=midominio,dc=com'
       -W -f ./passwd.ldif
       ```

5.5. Instalación de un cliente LDAP

Describiremos como como configurar un cliente para que acceda a la información almacenada en el directorio de LDAP del servidor

- Tres pasos:
  1. Indicar en el fichero `/etc/ldap/ldap.conf` la información sobre el servidor LDAP y el URI
  2. Instalar y configurar el *Name Service Switch* (fichero `/etc/nsswitch.conf`)
  3. Instalar y configurar el módulo de autenticación (PAM, *Pluggable Authentication Modules*)
- Los módulos necesarios para esta configuración pueden descargarse de la página de PADL (www.padl.com) o directamente como paquetes Debian
  - Descargar los paquetes `libnss-ldap`, `libpam-ldap` (opcionalmente, descargar también el `nscd`)
  - En la configuración indicar como URI `ldap://`*`ip_del_servidor`*`/`, el DN de la base del directorio LDAP, versión 3 de LDAP, y el password del administrador LDAP
  - Estas configuraciones se guardan en los ficheros `/etc/libnss-ldap.conf` y `/etc/pam_ldap.conf` (ver `man libnss-ldap.conf` y `man pam_ldap.conf`)
  - En la instalación nos pide la clave del administrador de LDAP:
    - esta clave se guarda en plano en los ficheros `/etc/pam_ldap.secret` y `/etc/libnss-ldap.secret`, y se usa para que el `root` del sistema pueda modificar directamente las contraseñas de los usuarios
    - si no se quiere tener ese fichero con la clave, no indicar ninguna clave cuando nos la pide y en los ficheros `/etc/libnss-ldap.conf` y `/etc/pam_ldap.conf` comentar la línea que empieza por `rootbinddn`
- En algunas distros (RedHat) existe la herramienta `authconfig` que facilita esta configuración

Configurar el *Name Service Switch*

El NSS se encarga, entre otras, de realizar la correspondencia entre los números y nombres de usuario

- permite gestionar los permisos de acceso de usuarios a ficheros
- se configura a través del fichero `/etc/nsswitch.conf`
- la configuración de ese fichero la vimos en el tema 3

Pasos (despues de haber instalado el paquete `libnss-ldap`):

1. Modificar las líneas de `passwd`, `group` y `shadow` del fichero `nsswitch.conf`

   ```
   passwd:    files ldap
   group:     files ldap
   shadow:    files ldap
   ```

   - esto indica al NSS que busque la información primero en los ficheros y, si no la encuentra, en el servidor LDAP

2. Probar que funciona:

   a) Crear el directorio `/home/pepe` y cambiarle el propietario a `pepe`

   b) Hacer un `su - pepe` para ver que podemos cambiar al usuario `pepe` en el cliente

Configurar el módulo de autenticación

Aunque el usuario es válido en el cliente, todavía no podemos autenticarnos (entrar en la cuenta)

- Debemos configurar el servicio PAM

PAM (*Pluggable Authentication Module*) biblioteca de autentificación genérica que cualquier aplicación puede utilizar para validar usuarios, utilizando por debajo múltiples esquemas de autentificación alternativos (ficheros locales, Kerberos, LDAP, etc.)

- permite añadir nuevos mecanismos de autentificación (Kerberos, LDAP,...) sin tener que modificar los servicios de entrada al sistema como `login`, `ftp`, `ssh`, etc.
- PAM utiliza módulos que proporcionan autentificación en los servicios de entrada al sistema:
  - Módulos de autentificación (`auth`): comprobación de contraseñas (utilizado por el proceso de `login` para averiguar si las credenciales tecleadas por el usuario (nombre y contraseña) son correctas)

        - Módulos de cuentas (`account`): controlan que la autenticación sea permitida (que la cuenta no haya caducado, que el usuario tenga permiso de iniciar sesiones a esa hora del día, etc.)
        - Módulos de contraseña (`password`): permiten cambiar contraseñas
        - Módulos de sesión (`session`): configuran y administran sesiones de usuarios (tareas adicionales que son necesitadas para permitir acceso, como el montaje de directorios, actualización del fichero `lastlog`, etc.)
    - Las librerías de PAM están en `/lib/security` y los ficheros de configuración en `/etc/pam.d/`
    - Existe un fichero de configuración para cada servicio que usa PAM
    - También existen ficheros comunes que son incluidos por los ficheros de configuración: `common-auth`, `common-account`, `common-password` y `common-session`
    - Versiones recientes de PAM gestionan estos ficheros mediante el comando `pam-auth-update` y los ficheros en `/usr/share/pam-configs`

Configuración de PAM para usar LDAP el proceso de instalación del paquete `libpam-ldap` ya modifica de forma adecuada los ficheros de configuración de PAM, pero da algunos problemas

1. En el fichero `/usr/share/pam-configs/ldap`, borra la opción `use_authtok` (con esta opción, no permite que los usuarios cambien su contraseña) y ejecuta `pam-auth-update`

2. Si queremos crear directorios home "al vuelo" (el home del usuario se crea al entrar por primera vez en su cuenta), crear el fichero `/usr/share/pam-configs/my_m` con el siguiente contenido

   ```
   Name: activate mkhomedir
   Default: yes
   Priority: 900
   Session-Type: Additional
   Session:
           required      pam_mkhomedir.so umask=0022 skel=/etc/skel
   ```

   y ejecutar `pam-auth-update` (Nota: el directorio solo se va a crear cuando el usuario se conecta en servidor)

3. Probar que funciona:

a) Entrar en la cuenta como un usuario LDAP

b) Cambiar la contraseña del usuario

Paquete `nscd` Se trata del *Name Service Cache Daemon*

- Hace caché para los nombres leidos del servidor LDAP para aumentar la eficiencia

5.6. Configuración de LDAP con múltiples servidores

Podemos configurar varios servidores LDAP que mantengan imágenes sincronizadas de la información del directorio

- equilibran la carga de las consultas, y mejora la tolerancia a fallos

Esquema de maestro único y múltiples esclavos

- el *maestro* mantiene la copia principal sobre la que se hacen los cambios
  - si un cliente intenta hacer un cambio en un esclavo, este lo redirige automáticamente al maestro
- cada vez que se produce un cambio en el directorio del maestro, el servicio `slapd` escribe dicho cambio, en formato LDIF, en un fichero de log
- el demonio `slurpd` lee dichos cambios e invoca las operaciones de modificación correspondientes en todos los esclavos

Para configurarlo debemos hacer que el maestro y los esclavos partan de un estado de directorio común

- copiar manualmente la base de datos LDAP del maestro a todos los esclavos

En el maestro y en los esclavos debemos modificar el fichero `/etc/ldap/slapd.conf`

- Maestro:
  - añadir una directiva `replica` por cada esclavo, donde se indique el nombre del esclavo y una cuenta con permiso de escrtitura en el LDAP del esclavo (`bindn`)

```
replica uri=ldap://esclavo.midominio.com:389
        binddn="cn=Replicator,dc=midominio,dc=com"
        bindmethod=simple credentials=CONTRASEÑA_PLANA
```

- indicar el fichero de log donde se guardan los cambios

```
replogfile   /var/lib/ldap/replog
```

- Esclavo:
  - añadir una línea `updatedn` indicando la cuenta con la que el servicio `slurpd` del servidor maestro va a realizar las modificaciones en la réplica del esclavo
  - esa cuenta debe tener permisos de escritura en la base de datos del esclavo, y debe coindidir con la indicada en el campo `binddn` del maestro
  - añadir una línea `updateref` para indicar al servidor esclavo que cualquier petición directa de modificación que venga de un cliente debe ser redireccionada al servidor maestro

```
updatedn   "cn=Replicator,dc=midominio,dc=com"
updateref  ldap://maestro.midominio.com
```

Para más detalles ver: OpenLDAP Administrator's Guide: Replication with slurpd

5.7. Herramientas de administración de LDAP

Administrar LDAP desde línea de comandos resulta muy engorroso

- Existen numerosas herramientas visuales que facilitan la gestión de LDAP
- Algunas de ellas son:
  1. phpldapadmin - interfaz basada en web para administrar servidores LDAP
  2. gosa - herramienta de administración, basada en PHP, para gestión de cuentas y sistemas en LDAP
  3. ldap-account-manager - webfrontend para gestión de cuentas en un directorio LDAP
  4. gq - cliente LDAP basado en GTK+/GTK2 (bastante simple)
  5. cpu - herramientas de gestión para consola: proporciona comandos tipo `useradd`/`userdel` para usar con LDAP

6. Compartición Linux-Windows: Samba

Samba permite a un sistema UNIX conversar con sistemas Windows a través de la red de forma nativa

- el sistema Unix aparece en el "Entorno de red" de Windows
- los clientes Windows pueden acceder a sus recursos de red e impresoras compartidas
- el sistema UNIX puede integrarse en un dominio Windows, bien como Controlador Primario del Dominio (PDC) o como miembro del dominio

Veremos una configuración "básica" de Samba; para saber más:

1. The Samba Homepage
2. The Official Samba-3 HOWTO and Reference Guide
3. The Unofficial Samba HOWTO
4. The Linux Samba-OpenLDAP Howto
5. Using Samba, 2nd Edition, O'Reilly & Associates
6. Integración de redes con OpenLDAP, Samba, CUPS y PyKota
7. Curso de Integración de Sistemas Linux/Windows

6.1. Funcionamiento de Samba

Samba implementa los protocolos NetBIOS y SMB

- NetBIOS: protocolo de nivel de sesión que permite establecer sesiones entre dos ordenadores
- SMB (*Server Message Block*): permite a los sistemas Windows compartir ficheros e impresoras (llamado *Common Internet File System*, CIFS por Microsoft)

Samba ofrece los siguientes servicios

- Servicios de acceso remoto a ficheros e impresoras
- Autenticación y autorización
- Servicio de resolución de nombres

Demonios Samba

Samba utiliza dos demonios: `smbd` y `nmdb`

- `smdb` permite la compartición de archivos e impresoras sobre una red SMB, y proporciona autentificación y autorización de acceso para clientes SMB; ofrece los dos modos de compartición de recursos existentes en Windows
  - modo basado en usuarios o modo *user* (propio de los dominios Windows NT o 2000)
  - modo basado en recursos o modo `share` (propio de Windows 3.11/95)
- `nmbd` permite que el sistema Unix participe en los mecanismos de resolución de nombres propios de Windows (WINS), lo que incluye
  - anuncio en el grupo de trabajo
  - gestión de la lista de ordenadores del grupo de trabajo
  - contestación a peticiones de resolución de nombres
  - anuncio de los recursos compartidos

Otras utilidades Samba

Adicionalmente a los dos programas anteriores, Samba ofrece varias utilidades; algunas de las más relevantes son:

- `smbclient` interfaz que permite a un usuario de un sistema Unix conectarse a recursos SMB y listar, transferir y enviar ficheros
- `swat` (*Samba Web Administration Tool*) utilidad para configurar Samba de forma local o remota utilizando un navegador web
- `smbfs` sistema de ficheros SMB para Linux: permite montar un recurso SMB ofrecido por un servidor SMB (un sistema Windows o un servidor Samba) en un directorio local
- `winbind` permite al sistema UNIX resolver nombres y grupos desde un servidor Windows

6.2. Instalación básica de Samba

Veremos una instalación básica de Samba en nuestro sistema Debian:

- permitirá desde un Windows acceder a los directorios de usuarios

Instalación de los paquetes

El paquete básico a instalar es `samba` que incluye los demonios de Samba

- instala también el paquete `samba-common`, que incluye utilidades como `smbpasswd` y `testparm`

Otros paquetes de Samba son:

- `smbclient` herramientas para el cliente Samba
- `smbfs` comandos para montar y desmontar `smbfs`
- `swat` *Samba Web Administration Tool*
- `winbind`

Sólo instalaremos `samba` y `samba-common`

- la instalación nos pide un nombre de Grupo de Trabajo/Dominio
  - indicar un nombre, que debemos usar en el sistema Windows

6.3. Configuración de Samba

La configuración de Samba se realiza en el fichero `/etc/samba/smb.conf`

- establece las características del servidor Samba, así como los recursos que serán compartidos en la red

Ejemplo sencillo:

```
[global]
  workgroup = MIGRUPO
[homes]
  comment = Home Directories
[pub]
  path = /espacio/pub
```

Estructura del archivo `smb.conf`

El fichero `/etc/samba/smb.conf` se encuentra divido en secciones, encabezados por una palabra entre corchetes

- En cada sección figuran opciones de configuración, de la forma `etiqueta = valor`, que determinan las características del recurso exportado por la sección

- Existen tres secciones predefinidas: `global`, `homes` y `printers`
- Otras secciones (como `pub` en el ejemplo anterior) definen otros recursos para compartir

Secciones predefinidas:

`[global]` define los parámetros de Samba a nivel global del servidor, por ejemplo, el programa utilizado para que un usuario pueda cambiar su clave (`passwd program`)

`[homes]` define automáticamente un recurso de red por cada usuario conocido por Samba; este recurso, por defecto, está asociado al directorio *home* del usuario en el ordenador en el que Samba está instalado

`[printers]` define un recurso compartido por cada nombre de impresora conocida por Samba

Niveles de Seguridad

Samba ofrece dos modos de seguridad, correspondientes a los dos modos de compartición de recursos ya vistos

- Modo *share*: cada vez que un cliente quiere utilizar un recurso ofrecido por Samba, debe suministrar una contraseña de acceso asociada a dicho recurso
- Modo *user*: el cliente establece una sesión con el servidor Samba (mediante usuario y contraseña); una vez Samba valida al usuario, el cliente obtiene permiso para acceder a los recursos ofrecidos por Samba

El nivel de seguridad se especifica con la opción `security`, la cual pertenece a la sección `[global]`

```
security = share | user | server | domain | ADS
```

Los niveles `user`, `server`, `domain` y `ADS` corresponden todos ellos al modo de seguridad `user`

- Nivel `user`: el encargado de validar al usuario es el sistema Unix donde Samba se ejecuta; es necesario que existan los mismos usuarios y con idénticas contraseñas en los sistemas Windows y en el servidor Samba
- Nivel `server`: Samba delega la validación del usuario en otro ordenador, normalmente un sistema Windows 2000 (método no recomendado)

- Nivel `domain`: el ordenador en el que se delega la validación debe ser un Controlador de Dominio (DC), o una lista de DCs; el sistema Samba actúa como miembro de un dominio
- Nivel `ADS`: en Samba-3 permite unirse a un dominio basado en Active Directory como miembro nativo

El modo por defecto es `user`

6.4. Otros comandos Samba

La suite Samba incluye otros comandos, como son:

- `testparm` permite chequear el fichero `smb.conf` para ver si es correcto
- `net` herramienta básica para administrar Samba y servidores SMB remotos; funciona de forma similar al comando `net` de DOS
- `smbpasswd` permite cambiar la contraseña usada en las sesiones SMB; si se ejecuta como root también permite añadir y borrar usuarios del fichero de contraseñas de Samba
- `smbstatus` muestra las conexiones Samba activas
- `smbclient` permite a un usuario de un sistema Unix conectarse a recursos SMB y listar, transferir y enviar ficheros