



Universidad de Murcia

Facultad de Informática

Departamento de Ingeniería y Tecnología de Computadores

Área de Arquitectura y Tecnología de Computadores

PRÁCTICAS DE SS.OO.

I.I./I.T.I. SISTEMAS/I.T.I. GESTIÓN

Práctica 1 – El Sistema Operativo Linux

OCTUBRE DE 2006

Índice

1. UN POCO DE HISTORIA	3
1.1. Características de Linux	4
1.2. Distribuciones	4
1.3. Direcciones de Internet relacionadas con Linux	6
2. PRIMEROS PASOS	7
3. FICHEROS	8
3.1. Comodines	9
3.2. Directorios	9
3.3. Ficheros Especiales y Dispositivos	10
4. PROCESOS	11
5. IMPRESIÓN	13
6. DISCOS	14
7. ÓRDENES Y PROGRAMAS ÚTILES	14
7.1. La orden find	14
7.2. La orden tar	15
7.3. Otras órdenes y programas de uso común	15
7.4. Conexión remota y transferencia de ficheros: telnet , ftp , y ssh	16
8. X-WINDOW	16
8.1. Breve historia de X-Window	16
8.2. X-Window System	17
8.3. Aplicaciones para X-Window System	17
9. EJERCICIOS	18

1. UN POCO DE HISTORIA

UNIX surgió alrededor de 1970 como proyecto personal de Ken Thompson que trabajaba para Bell Labs de AT&T. Inicialmente, Thompson programó UNIX en una vieja PDP-7. Debido a su éxito, pronto se le unió Dennis Ritchie y más tarde el resto de miembros de su departamento. A continuación, portaron UNIX a varios modelos de la familia PDP y, Thompson y Ritchie volvieron a escribir UNIX en C, lo que posteriormente facilitaría su portabilidad a otras máquinas. En 1974 Ritchie y Thompson escribieron un artículo relativo a UNIX cuya publicación estimuló a muchas universidades a pedir a Bell Labs una copia de UNIX, incluido el código fuente. La versión que se convirtió en el primer estándar del mundo académico fue la Versión 6, llamada así porque era descrita en la sexta edición del manual del programador de UNIX.

En 1984, tras la separación de Bell Labs y AT&T, AT&T lanzó su primera versión comercial de UNIX, el Sistema III, que no fue bien recibido, por lo que un año más tarde se reemplazó por una versión mejorada, el Sistema V, del que sacó las versiones 2, 3 y 4.

Una de las primeras universidades que adquirió pronto la Versión 6 de UNIX fue la Universidad de California en Berkeley. Puesto que disponía de todo el código fuente, Berkeley hizo substanciales mejoras y produjo sus propias versiones denominadas BSD. Entre las mejoras realizadas estaban la paginación, un sistema de ficheros más rápido y con nombres de fichero más largos, una pila de protocolos de red (TCP/IP) y numerosas utilidades. Todas estas mejoras hicieron varios vendedores como basaran su versión de UNIX en BSD.

A finales de la década de los ochenta, se utilizaban dos versiones de UNIX distintas e incompatibles en ciertos aspectos: 4.3BSD y la versión 3 del Sistema V. El primer intento serio de reconciliar las dos versiones de UNIX, bajo los auspicios del IEEE Standards Board dio lugar al proyecto POSIX (POS por Portable Operating System y IX por unIX). El comité POSIX produjo una serie de estándares (del 1003.0 al 1003.10) siendo el 1003.1 (también denominado POSIX.1) el más importante de ellos y que definía el conjunto de llamadas al sistema que cualquier sistema UNIX debía proporcionar.

En 1993 Novell compró UNIX System Laboratories (USL) de AT&T y con ello los derechos de la marca UNIX. La última versión de UNIX de USL fue el UNIX Sistema V Versión 4.2 (SVR4.2). USL, con la participación con Novell, produjo una versión comercial del SVR4.2 denominada UnixWare. En 1995 Novell vendió su versión de UNIX a Santa Cruz Operation (SCO). Posteriormente, en el año 2001, Caldera Systems compró SCO y la nueva compañía pasó a llamarse Caldera International, para cambiar de nuevo de nombre en el año 2002 y convertirse en The SCO Group. Otras muchas compañías han desarrollado sus propias versiones de UNIX. La de Microsoft se denomina XENIX. The SCO Group continúa comercializando su propia versión denominada SCO UnixWare. La de IBM se denomina AIX. Y Sun Microsystems ha contribuido enormemente a la popularidad de UNIX promocionando el SunOS, que sustituyó hace pocos años por su Solaris.

En 1991, en medio de esta guerra comercial, surgió Linux. Linux es la idea original de Linus Torvalds, un estudiante de informática de la universidad de Helsinki que por aquel entonces tenía tan solo 23 años. Su idea era mejorar Minix, una versión de UNIX creada por Andrew S. Tanenbaum con fines educativos. A partir de Minix, creó un sistema operativo rudimentario y puso el código fuente a disposición de cualquiera. Pronto se le unieron diversos programadores de todo el mundo a través de Internet que desinteresadamente aportaban su trabajo para mejorar Linux. En muy poco tiempo, Linux adquirió todas las características de cualquier UNIX comercial.

No obstante, un sistema operativo no es sólo un núcleo. Se necesita un conjunto de aplicaciones con las que llevar a cabo un trabajo útil. Estas aplicaciones las aportó el proyecto GNU de la *Free Software Foundation* (FSF). El proyecto GNU pretende crear una versión completa de UNIX gratuita. Tenía ya varias aplicaciones como un compilador de C (*gcc*), un potente editor de texto (*emacs*) y un *shell* (*bash*), pero carecía de un sistema operativo, y Linux vino a llenar ese hueco. En este caso, las aplicaciones no se adaptaron al sistema operativo sino que Linux se adaptó para que dichas aplicaciones funcionaran sobre él. Esta decisión pragmática hizo que de la noche a la mañana surgieran gran número de aplicaciones que funcionaban en Linux como, por ejemplo, un sistema X-Windows proporcionado por el proyecto XFree86.

En marzo de 1994 apareció la versión 1.0 del núcleo y a finales de 1996 la versión 2.0. Actualmente, la versión del núcleo de Linux es la 2.6.18 (septiembre de 2006). Téngase en cuenta que las versiones con el segundo número impar son versiones de desarrollo, es decir, versiones en las que se están añadiendo mejoras y de las que no se asegura un funcionamiento correcto. En la actualidad, Linux se utiliza en entornos muy diversos, desde proveedores de Internet, hasta grandes empresas corporativas, pasando por universidades y órganos gubernamentales.

1.1. Características de Linux

Algunas de las características más destacables de Linux (algunas de ellas compartidas con otros UNIX) son:

- **Multiplataforma:** funciona en múltiples arquitecturas tanto de 32 como de 64 bits (Intel, AMD, PowerPC, ARM, . . .), es decir, Linux es independiente de la arquitectura. Su portabilidad se debe a que está escrito en lenguaje C y contiene muy poco código ensamblador.
- **Multitarea:** permite ejecutar varios procesos al mismo tiempo porque soporta aplicaciones multihilo (*multithread*), arquitecturas SMP (*Symmetric Multiprocessing*) y procesadores *multicore*.
- **Multiusuario:** varios usuarios pueden trabajar en la misma máquina al mismo tiempo.
- **Protección:** el bloqueo de un proceso no afecta al resto del sistema (memoria virtual con paginación).
- **Consolas virtuales:** múltiples sesiones pueden coexistir en la misma máquina al mismo tiempo de forma que el usuario puede pasar de una a otra con la combinación de teclas ALT+Fx donde Fx es F1,2,3,4,5,6.
- **Sistema de ficheros transaccional** `ext3` que permite nombres de ficheros de hasta 255 caracteres y tamaños de fichero de hasta 8 TB. A diferencia de otros sistemas de ficheros, no requiere herramientas de defragmentación porque minimiza la fragmentación de la información en disco. Soporta también otros sistemas de ficheros transaccionales como XFS, JFS y ReiserFS, y también los sistemas de ficheros nativos de otros sistemas operativos: MS-DOS (FAT16), Windows 98 (VFAT y FAT32), Windows NT/2000/2003/XP (NTFS), OS/2 (HPFS), MacOS (HFS), CD-ROM (ISO9660 con extensiones *Rock Ridge* y *Joliet*), Minix, etc.
- Soporte para **redes TCP/IP con IPv4 e IPv6** (protocolos usados en Internet), IPX/SPX (protocolos de las redes Novell NetWare), NetBEUI (protocolo de las redes Windows para Trabajo en Grupo y Windows 95/98), SMB/CIFS/Samba (protocolo para compartir archivos e impresoras), Appletalk (protocolo de las redes Apple para Macintosh), PPP (protocolo de punto a punto para el acceso a Internet con módem/ADSL), y PLIP (protocolo de puertos paralelos). Además, una máquina que ejecuta Linux puede actuar como *router*, *firewall*, *proxy*, etc.
- Ejecución de **ficheros binarios con formato ELF** (nativo), y con formatos pertenecientes a otros sistemas operativos tipo UNIX, mediante **carga por demanda**, es decir, sólo se leen de disco aquellas partes del programa que están siendo usadas en ese momento. También existe la posibilidad de utilizar **librerías compartidas** tanto estáticas como dinámicas (similares a las DLLs de Windows).
- **Open Source:** el código fuente de Linux está disponible y se distribuye mediante licencia GPL, lo que significa que se pueden hacer tantas copias como se desee y distribuir las libremente.

1.2. Distribuciones

Todo sistema operativo está constituido por dos elementos esenciales:

- El núcleo o *kernel* que define las características del sistema operativo.
- Los programas del sistema que son todos aquellos programas que forman parte del sistema operativo y que se ejecutan sobre el núcleo.

Una distribución de Linux está formada por el núcleo de Linux y por un conjunto de programas que lo acompañan. Las distintas distribuciones se diferencian en aspectos como:

- Métodos de instalación y configuración.
- Programas del sistema y aplicaciones/utilidades.
- Gestor de ventanas utilizado por defecto.

Existen muchas distribuciones de Linux. Algunas de ellas son:

- **Fedora Core (Fedora Linux):** distribución mantenida por la comunidad Fedora Project y promovida por la empresa norteamericana Red Hat Software, Inc.. Surgió en 2003 cuando Red Hat decidió marcar claramente las diferencias entre su versión libre y su versión de pago (Red Hat Enterprise Linux). Su instalación es sencilla, y la actualización de software, así como la instalación y desinstalación de aplicaciones, es rápida y eficiente gracias a su sistema de paquetes con formato RPM.
- **Debian GNU/Linux:** distribución mantenida por voluntarios de todo el mundo (Debian Project) siguiendo el espíritu de Linux y del proyecto GNU. **Ubuntu:** distribución basada en Debian patrocinada por la empresa sudafricana Canonical Ltd. Su instalación, configuración, actualización, y uso son extremadamente sencillos gracias a su integración con el escritorio GNOME, y a su potente sistema de paquetes con formato DEB, que está completamente automatizado con la herramienta `apt-get`.
- **SUSE Linux:** distribución desarrollada por la empresa alemana del mismo nombre que en 2004 fue absorbida por Novell Netware. Aunque no guarda relación alguna con Red Hat, SUSE utiliza paquetes con formato RPM y ofrece una versión de pago denominada SUSE Linux Enterprise Edition. Su instalación, configuración y actualización son muy fáciles gracias a una herramienta denominada YaST2.

No vamos a explicar la instalación de ninguna distribución en concreto ya que el proceso varía de una a otra. Sin embargo, sí vamos a repasar algunos requisitos que son comunes a todas las distribuciones:

- Todas ellas pueden coexistir con otros sistemas operativos en el mismo sistema, en un disco duro independiente o en el mismo disco duro. Si tenemos un disco duro independiente, podemos instalar Linux directamente, ya que habitualmente el propio proceso de instalación incluye herramientas para particionar el disco. Si tenemos una única partición, tenemos que reducir el tamaño de dicha partición para poder crear las nuevas particiones requeridas por Linux. Para hacer esto, necesitamos una herramienta como Partition Magic o similar que permite realizar la tarea sin perder los datos que ya tenemos. En todo caso, el proceso podría corromper los datos y/o el sistema operativo residentes en las partición original por lo que resulta recomendable realizar una copia de seguridad antes de iniciarlo.
- La instalación de Linux requiere normalmente dos particiones. Una partición para albergar el sistema operativo y los datos de los usuarios (podría crearse más de una), y una partición de intercambio o `swap`. La partición de intercambio es una partición que se utiliza única y exclusivamente para poder realizar el intercambio de página requerido por el mecanismo de memoria virtual. Esta partición debe crearse independientemente de la cantidad de memoria RAM instalada en el sistema y, como regla empírica, su tamaño debe ser el doble del tamaño de aquella. En lugar de utilizar una partición de intercambio podemos crear también un fichero de intercambio (como hace Windows) pero el rendimiento será peor.
- Todas las distribuciones suelen ofrecer la opción de realizar una instalación mínima que incluye un conjunto de paquetes clasificados como *requeridos* y que son imprescindibles para el correcto funcionamiento del sistema. Al realizar la instalación por primera vez, conviene elegir esta opción ya que siempre habrá tiempo de añadir más paquetes cuando los necesitemos.
- Al terminar el proceso de instalación, es necesario seleccionar el método de arranque de Linux. Básicamente existen dos métodos:
 - Crear un disquete de arranque desde el que se iniciará el ordenador.
 - Utilizar un cargador o *loader* que se ejecuta al arrancar el ordenador que nos permite seleccionar un sistema operativo de entre los que tenemos instalados. Los cargadores más utilizados en el mundo Linux se denominan LILO y GRUB. Ambos se pueden instalar en el MBR (*Master Boot Record*) del disco duro o en la partición de Linux. Sea cual sea el cargador elegido, si se instala en el MBR, hay que tener en cuenta que Windows destruye el MBR al instalarse, haciendo imposible el arranque de Linux. La solución pasa por acceder a la partición de Linux mediante un disquete y volviendo a instalar el cargador elegido en el MBR.

1.3. Direcciones de Internet relacionadas con Linux

General

- <http://www.linux.org>: Linux Online.
- <http://loll.sourceforge.net>: Loads of Linux Links.
- <http://www.linuxiso.org>: Imágenes de varias distribuciones.
- <http://www.hispalinux.es>: Asociación HispaLinux.
- <http://www.diariolinux.com>: Wiki sobre Linux en español.
- <ftp://ftp.rediris.es/pub/linux>: imágenes de varias distribuciones y software para Linux en general.

Software

- <http://linux.tucows.com>: Versión de Tucows para Linux.
- <http://rpmfind.net>: Aplicaciones y utilizadas en forma de paquetes RPM. Incluye un motor de búsqueda.
- <http://www.freshmeat.net>: Aplicaciones y utilidades clasificadas por categorías. Incluye un motor de búsqueda.

Distribuciones

- <http://www.fedora.redhat.com>: Fedora Core
- <http://www.fedora-es.com>: Fedora Core (español)
- <http://www.redhat.com>: Red Hat Enterprise Linux
- <http://www.redhat.es>: Red Hat Enterprise Linux (español)
- <http://www.debian.org>: Debian GNU/Linux (español)
- <http://www.ubuntu.com>: Ubuntu - Live CD e instalación
- <http://www.ubuntu-es.org>: Ubuntu - Live CD e instalación (español)
- <http://www.opensuse.org>: SUSE Linux
- <http://www.novell.com/es-es/linux/suse>: SUSE Linux Enterprise
- <http://www.knoppix.org>: KNOPPIX - Live CD

Documentación

- <http://www.kernel.org>: Información sobre el núcleo de Linux
- <http://www.tldp.org>: The Linux Documentation Project
- <http://es.tldp.org>: The Linux Documentation Project (español)
- <http://www.insflug.org>: Proyecto de traducción de HOWTOs y FAQs (COMOs y PUFs)

2. PRIMEROS PASOS

Para poder trabajar con Linux es necesario disponer de una cuenta. Dicha cuenta está compuesta por un nombre de usuario (**login**) y una contraseña (**password**) que se utilizan para verificar la identidad del usuario.

Una vez iniciada una sesión (tras indicar un login y un password correctos) nos encontramos con el indicador del sistema (**prompt**). Dicho `prompt` suele terminar en `$` para un usuario normal y en `#` para el superusuario. El superusuario (**root**) administra el sistema y, por tanto, no tiene ninguna limitación pudiendo hacer cualquier cosa en el sistema. Generalmente el superusuario se encarga de la configuración, instalación y actualización de software, y de las tareas de configuración y mantenimiento del sistema.

Al iniciar una sesión, el directorio en el que nos encontramos se conoce como directorio de trabajo o directorio **home** (a dicho directorio se le puede hacer también referencia en cualquier orden usando la abreviatura `~`). En cualquier momento podemos saber el directorio en el que nos encontramos con la orden **pwd**. Nótese que en Linux se utiliza la barra `/` en las rutas de ficheros y directorios, y no la barra `\` como ocurre en Windows. Para cambiar de directorio, usamos la orden **cd** `<directorio>`. Si ejecutamos **cd** sin argumentos, regresamos a nuestro directorio de trabajo o home. Y si ejecutamos **cd -** regresamos al último directorio en el que nos encontrábamos antes de cambiar al directorio actual.

Podemos averiguar qué otros usuarios están conectados en nuestra máquina al mismo tiempo con la orden **who**. También existe la orden **whoami** que nos dice quiénes somos nosotros. Otra orden útil relacionada es **w** que muestra qué usuarios se encuentran conectados, desde qué terminal o sistema remoto están conectados, qué orden están ejecutando en ese momento, etc.

En general, la estructura de las órdenes suele ser `orden opciones parámetros`. Las opciones se especifican con uno o dos guiones, como `-n`, `-d` o `--help`, y habitualmente pueden agruparse, es decir, en lugar de poner `-n -d`, podemos poner `-nd` o `-dn`. Además, es importante tener en cuenta que Linux diferencia entre mayúsculas y minúsculas por lo que `LS` es distinto de `ls`.

Cuando no conocemos muy bien el funcionamiento de una orden, podemos obtener ayuda con **orden --help** u **orden -?**. Si ninguno de estos dos métodos funciona, podemos consultar la página de manual de dicha orden con la orden **man** (más información con `man man`). Las páginas de manual están estructuradas en secciones (la sección 1 corresponde al intérprete de órdenes, la sección 2 a llamadas al sistema, la sección 3 a llamadas de la biblioteca del sistema, etc.). Esta orden se suele utilizar de tres maneras:

- **man <tema>**: cuando el tema que queremos consultar aparece en una única sección. Ejemplo: `man ls`.
- **man -a <tema>**: cuando el tema puede aparecer en varias secciones. Ejemplo: `man -a mount`.
- **man -s<sección> <tema>**: cuando conocemos la sección del tema. Ejemplo: `man -s8 mount`.

Si las páginas del manual no son suficientes, podemos consultar el directorio `/usr/share/doc` que suele contener gran cantidad de información, o utilizar el sistema de información `info`, mediante **info <tema>**. Por último, decir que para consultar la ayuda relativa a *comandos internos* del intérprete de comandos (es decir, que no están en ficheros binarios ejecutables aparte, sino que se encuentran en el propio código del intérprete `bash`), se usa la orden **help <comando>**.

Para terminar esta sección, vamos a aprender otra característica que se suelen utilizar con mucha frecuencia. En el intérprete de órdenes (*shell*) más común, `bash`, **el tabulador nos permite completar líneas**. Por ejemplo, si existe un fichero de texto llamado "Ejemplo.txt" y queremos leer su contenido, podemos escribir `less Eje<TAB>` y, automáticamente, se completará la línea quedando como `less "Ejemplo.txt"`. Si existe ambigüedad (hay dos o más ficheros que empiezan exactamente igual), no se podrá completar la línea. En este caso, pulsado el tabulador una segunda vez obtendremos una lista con todas las posibilidades.

También existe una **historia de órdenes** que podemos consultar con las teclas de cursor arriba y cursor abajo. Incluso podemos escribir expresiones como **!orden**, lo que nos ejecutará la última orden `orden tal` y como la hayamos escrito (incluyendo opciones y argumentos). Además, también es posible concatenar varias órdenes con `;` en una misma línea de órdenes, por ejemplo, `clear; ls`.

En una misma consola física (el teclado y la pantalla que se encuentra físicamente conectados al ordenador) es posible tener varias sesiones abiertas al mismo tiempo. Cada una de ellas es lo que se conoce como consola o

terminal virtual. Podemos cambiar de una **consola virtual** a otra con la combinación de teclas **ALT+Fx** donde F1 corresponde a la primera terminal virtual, y así sucesivamente.

El ratón se puede utilizar para copiar texto en terminal (moviéndolo al mismo tiempo que se tiene pulsado el botón izquierdo), y pegarlo en la posición del cursor (de la misma terminal virtual o de otra distinta) pulsando el botón derecho.

Por último, para cerrar la sesión, ejecutamos **exit**, **logout**, o pulsamos **CTRL+D**.

3. FICHEROS

En el sistema de ficheros de Linux, denominado `ext3`, el nombre de un fichero puede tener hasta 255 caracteres de longitud y, si queremos que contenga espacios en blanco, debemos utilizar comillas (simples o dobles) para escribir su nombre.

La orden **ls** nos permite ver los nombres de todos los ficheros y directorios del directorio actual (o del que especifiquemos como parámetro). Esta orden admite muchas opciones, siendo una de las más importante la opción **-l** que genera una salida con el siguiente formato:

```
Tipo/permisos  enlaces  propietario  grupo  tamaño  fecha  hora  nombre
-rwxr-xr-x    1         antonio     users  138     Apr 5   19:34  test
```

El primer campo contiene 10 caracteres. El primero de ellos indica el tipo de fichero: `-` para ficheros regulares (normales), `d` para directorios, `c` (b) para ficheros que representan dispositivos de caracteres (bloques), `l` para enlaces simbólicos, etc. Los siguientes 9 caracteres se subdividen en 3 campos, cada uno de ellos indicando los permisos sobre ese fichero del propietario del fichero (`antonio`), del grupo al que pertenece el fichero (`users`) y del resto de usuarios. En nuestro ejemplo, el propietario puede leer, modificar el contenido o ejecutar el fichero (`rwx`), mientras que el grupo y el resto de usuarios sólo pueden leerlo y ejecutarlo (`r-x`). El número de enlaces que aparece es el de enlaces físicos (este concepto se estudia en teoría).

Algunas opciones útiles de la orden `ls` son:

- C**: mostrar el contenido en columnas.
- a**: mostrar también los nombres que empiezan por “.”.
- F**: añadir un carácter “/” a los directorios y un carácter “*” a los ejecutables.
- R**: mostrar el directorio actual y sus subdirectorios.
- i**: mostrar el número de nodo-i del fichero.
- d**: listar los directorios sin mostrar su contenido.

A continuación describimos otras órdenes relacionadas con ficheros:

- **cp** [**-i**] **<ficheros>** **<fichero>** | **<directorio>**. Copia uno o varios ficheros. Si el destino es un fichero, no se pueden utilizar comodines en el origen. La opción `-i` nos pide confirmación de sobrescritura en caso de que el destino exista. Se deben especificar obligatoriamente tanto el origen como el destino.
- **mv**. Similar a `cp` pero para mover y/o cambiar de nombre. Nota: véase también **rename**.
- **ln**. Sigue las mismas reglas que `cp`. Sirve para crear enlaces físicos. La opción `-s` se utiliza para crear enlaces simbólicos.
- **rm** [**-i** **-r**] **<ficheros>**. Borra ficheros. Un fichero borrado ya no se puede recuperar. La opción `-i` nos pide confirmación. La opción `-r` se utiliza para borrar un directorio y todo lo que contenga.

- **cat** <fichero> [<fichero> ...]. Visualiza y concatena varios ficheros.
- **chmod** [ugoa][+|=]rwx <ficheros>. Permite cambiar los permisos de un fichero para el propietario (u), para el grupo (g), para el resto de usuarios (o) o para todos (a). El + añade un permiso, el - lo elimina y el = fija como único permiso el indicado. Por ejemplo, `chmod g=w texto.txt` hace que el único permiso para el grupo sea el de escritura mientras que `chmod g+w texto.txt` hace que se añada el permiso de escritura para el grupo.
- **chown** <nuevo_propietario> <ficheros>. Permite al propietario de un fichero asignarle otro usuario diferente como propietario, es decir, cederle la propiedad. Nota: esta orden sólo la puede ejecutar el superusuario.
- **chgrp**. Similar a `chown` pero para cambiar el grupo. Nota: también se pueden cambiar el propietario y el grupo a la vez usando `chown <propietario>:<grupo> <fichero>`.

3.1. Comodines

Muchas órdenes de Linux pueden actuar sobre varios ficheros a la vez. Estos ficheros los podemos especificar en la línea de órdenes uno a uno, o podemos especificar un patrón al que se ajusten todos los nombres de los ficheros sobre los que queremos que actúe la orden.

Algunos caracteres especiales utilizados para especificar tales patrones son:

- “*”: 0 más caracteres cualesquiera.
- “?”: exactamente un carácter cualquiera.
- “[]”: sólo uno de los caracteres especificados entre corchetes.
- “[!]”: cualquier carácter siempre que no sea uno de los especificados entre corchetes tras el símbolo de exclamación.
- “{palabra, palabra, ...}”: cualquier palabra de la lista.

Ejemplos:

- “*bc”: todos los ficheros terminados en “bc”, como “abhjbc”.
- “[ab]h?c”: todos los ficheros de 4 letras cuyo primer carácter sea o bien “a” o bien “b”, el segundo carácter sea “h”, el tercer carácter sea cualquiera y el cuarto carácter sea “c”, como “ahoc” y “bhic”.

3.2. Directorios

En Linux, los directorios son ficheros especiales, por lo que muchos de los conceptos que hemos visto para ficheros también se pueden aplicar a los directorios. En el caso de los directorios, los permisos tienen un significado especial:

- r**: permiso para examinar el contenido del directorio, por ejemplo, con `ls`.
- w**: permiso para modificar el contenido del directorio, por ejemplo, para crear nuevos ficheros.
- x**: permiso para cambiar a dicho directorio con la orden `cd` y pasar a través de él para acceder a sus subdirectorios, por ejemplo, para realizar una búsqueda.

Los directorios se crean con `mkdir` y se borran con `rmdir` (o `rm -r`). La estructura típica del árbol de directorios en un sistema Linux se muestra a continuación. El directorio `/sbin` contiene programas que sólo pueden ser ejecutados por el superusuario. El directorio `/bin` contiene las órdenes más básicas disponibles para todos los usuarios. El directorio `/etc` contiene muchos ficheros de configuración. El directorio `/home` suele contener un directorio de trabajo para cada uno de los usuarios del sistema (excepto para el superusuario).

cuyo directorio de trabajo es `/root`). En el directorio `/usr` se suelen encontrar el resto de órdenes y todas las aplicaciones del sistema. Y, por último, el directorio `/var` suele contener muchos ficheros temporales y ficheros de registro que cambian de tamaño con frecuencia.

```
/
|-- bin
|-- dev
|-- etc
|-- home
|-- lib
|-- lost+found
|-- proc
|-- sbin
|-- tmp
|-- usr
|   |-- X11R6
|   |-- bin
|   |-- doc
|   |-- include
|   |-- lib
|   |-- local
|   |-- man
|   |-- sbin
|   |-- share
|-- var
|   |-- log
|   |-- mail
|   |-- spool
|       |-- cron
|       |-- lpd
|       |-- mail
```

3.3. Ficheros Especiales y Dispositivos

En Linux, al igual que ocurre con los directorios, los dispositivos se representan mediante ficheros especiales. Básicamente, existen dos tipos de ficheros especiales: los que representan a dispositivos de caracteres (como un puerto paralelo o un puerto serie), cuyo tipo en los permisos es `c`, y los que representan a dispositivos de bloques (como un disquete, una partición de un disco duro o un disco duro completo), cuyo tipo en los permisos es `b`. Todos estos ficheros especiales se encuentran en el directorio `/dev`.

Algunos ejemplos de ficheros especiales son:

- `/dev/fd0`: representa el disquete de la primera disquetera (A: en Windows).
- `/dev/hda1`: representa la primera partición del primer disco duro (C: en Windows).
- `/dev/hda2`: representa la segunda partición del primer disco duro.
- `/dev/hdb1`: representa la primera partición del segundo disco duro.
- `/dev/sda1`: representa a la primera partición del primer disco SCSI (otros dispositivos como los discos USB utilizan emulación SCSI por lo que aparecerán también como `/dev/sdaX`).
- `/dev/tty1`: representa la primera consola virtual.
- `/dev/tty2`: representa la segunda consola virtual.

- `/dev/pts/1`: representa la primera pseudo-terminal.
- `/dev/lp0`: representa el primer puerto paralelo.

Estos nombres se utilizan para realizar operaciones sobre los dispositivos que representan. En el caso de los dispositivos de bloques, las operaciones más típicas son:

- Dar formato a un disquete: `fdformat /dev/fd0`.
- Crear un sistema de ficheros:
 - `mkfs.msdos /dev/fd0`, `mkfs -t msdos /dev/fd0` o `mkdosfs /dev/fd0` para crear un sistema de ficheros de MS-DOS (FAT12) en un disquete.
 - `mkfs.ext3 /dev/hda3` o `mkfs -t ext3 /dev/hda3` para crear un sistema de ficheros nativo de Linux en la tercera partición del primer disco duro.
- Montar un sistema de ficheros:
 - `mount /dev/fd0 /mnt` monta el primer disquete en el directorio `/mnt`
 - `mount -t iso9660 /dev/cdrom /cdrom` montamos el CD-ROM en el directorio `/cdrom` especificando además el tipo del sistema de ficheros que contiene.
- Comprobar la consistencia: `fsck.ext2 /dev/hda4`, `fsck -t ext2 /dev/hda4` o también `e2fsck /dev/hda4` para realizar esta operación siendo necesario que la partición a comprobar no esté montada o que esté montada como de sólo lectura.
- Particionar un disco duro: `fdisk /dev/hdb` para particiona el segundo disco duro (nótese que no se indica ninguna partición concreta).

4. PROCESOS

Todos los procesos en Linux se pueden considerar formados por dos partes: una parte que se ejecuta en modo usuario y otra parte que se ejecuta en modo núcleo. La primera parte la constituye el código del programa en sí (y el código de las posibles funciones de librería que utilice). La segunda parte la constituye el código del propio sistema operativo. Al ejecutar un programa, se crea un proceso (o varios) que lo representa.

La mayoría de los procesos tienen una **entrada estándar** (por defecto, el teclado), una **salida estándar** (por defecto, la pantalla) y una **salida de error estándar** (por defecto, también la pantalla). La **redirección** de cualquiera de estos tres elementos se hace de forma similar a como se hace en Windows: “<<” para redirigir la entrada, “>>” para redirigir la salida, “>>” para redirigir la salida (pero concatenándola con el contenido actual del fichero usado), “2>>” (que no existe en Windows) para redirigir la salida de error y “2>>>” (que es equivalente a “>>>”, pero aplicado a la salida de error). También podemos utilizar una **tubería** (“|”) para conectar la salida estándar de una orden con la entrada estándar de la orden siguiente.

En el intérprete de órdenes, un proceso se puede ejecutar en **primer plano** (hasta que no termine su ejecución no podemos introducir la siguiente orden) o en **segundo plano** (mientras su proceso se ejecuta, podemos seguir ejecutando otras órdenes). Una orden se ejecuta en segundo plano poniendo un carácter “&” al final. Por ejemplo, al ejecutar

```
$ cat /etc/passwd > passwd.stdout 2> pass.stderr &
[1] 1544 (PID del proceso creado)
$
```

el número que aparece (1544) es el PID (*Process IDentification*) del proceso creado que se está ejecutando en segundo plano.

También es posible pasar a segundo plano un proceso que se está ejecutando en primer plano. Para ello, se pulsa CTRL+Z para detener la ejecución del proceso, y después se ejecuta **fg** para reanudar la ejecución del proceso en primer plano otra vez, o **bg** para reanudarlo en segundo plano. La orden **jobs** nos permite ver todos los trabajos suspendidos o en segundo plano.

La orden **ps** se utiliza para mostrar información acerca de los procesos existentes en el sistema. Por defecto, muestra información de los procesos asociados con nuestra sesión, aunque con las opciones adecuadas podemos ver todos los procesos de todos los usuarios. La salida de esta orden se ajusta al siguiente formato:

```
PID TTY TIME COMMAND
```

PID es el identificador del proceso. TTY es el terminal del que lee y escribe el proceso. TIME es el tiempo de ejecución (uso de CPU) del proceso. COMMAND es la orden que dio lugar a la creación del proceso.

Algunas opciones útiles de la orden **ps** son:

-l: muestra información más completa siguiendo el formato

```
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
```

Los nuevos campos tienen el siguiente significado. UID: identificación del usuario al que pertenece el proceso. PPID: PID del proceso padre del proceso. PRI: prioridad. NI: valor *nice*. SZ: tamaño, en páginas, de la imagen del proceso incluyendo código (texto), datos y pila. WCHAN: nombre de la función del kernel dónde duerme el proceso.

-aux: proporciona información de todos los procesos que se están ejecutando en el sistema (tengan asociada una terminal o no) junto con el nombre del usuario al que pertenece cada uno de ellos. Además, informa del porcentaje de CPU consumido por el proceso (%CPU), el porcentaje de memoria que ocupa (%MEM), el estado del proceso (STAT: véase PROCESS STATE CODES en la página del manual), tamaño en KBs de la memoria física ocupada por el proceso (RSS) y la hora de creación del proceso (START).

Cuando terminamos nuestra sesión, todos los procesos que se han creado durante esa sesión también finalizan. A veces, nos puede interesar terminar una sesión y dejar un proceso ejecutándose. Esto se consigue con la orden **nohup**:

```
$ nohup cat /etc/passwd &
$ nohup cat /etc/passwd | nohup wc > wc_passwd.stdout &
```

En el segundo caso, si se utiliza una tubería con varios órdenes, debemos colocar la orden **nohup** en cada orden de la tubería. Como una vez cerrada una sesión no hay terminal asociada a los procesos, si no se redirecciona la salida estándar ni la salida estándar de error, se crea un fichero para ambas salidas llamado **nohup.out**.

En algunas ocasiones, un proceso puede fallar y no terminar, y en otras podemos desear terminar un proceso antes de que concluya. Para ello, existe la orden **kill**:

- **kill 4314**: envía una señal número 15 al proceso cuyo PID es 4314 (indica al proceso que debe terminar su ejecución).
- **kill -9 4314**: envía una señal número 9 al proceso cuyo PID es 4314 (terminación incondicional).

En realidad, la orden **kill** se utiliza para enviar una señal cualquiera a un proceso, no sólo para terminarlo. No obstante, únicamente el superusuario puede enviar una señal a cualquier proceso. El resto de usuarios sólo pueden enviar señales a los procesos que les pertenecen.

El programa **top** es similar a la orden **ps** pero, en este caso, muestra información general sobre procesos, carga del sistema, y uso de memoria en tiempo real.

5. IMPRESIÓN

La impresión en Linux se realiza con la orden **lpr**. Esta orden coloca un trabajo en la cola de impresión y es el demonio de impresión (el proceso `lpd`) el que finalmente lo envía a la impresora. La orden `lpr` se puede utilizar de dos formas:

- Con una tubería: `cat trabajo.txt | lpr`
- Con el nombre del fichero a imprimir como parámetro: `lpr trabajo.txt`

Algunas opciones útiles de la orden `lpr` son:

- `-P <impresora>`: impresora a la que debe enviarse el trabajo.
- `-# <num-copias>`: número de copias del trabajo.

Por ejemplo, para imprimir la página de manual de `ls` en la impresora HP5M:

```
man -t ls | lpr -P HP5M
```

Una orden relacionada con `lpr` es **pr** que permite dar formato a lo que se quiere imprimir (que debe ser texto ASCII). Entre otras cosas, permite hacer lo siguiente:

- Numerar las páginas.
- Especificar un encabezado.
- Especificar la longitud de la página.
- Especificar el ancho de la página.
- Modificar la indentación.
- Imprimir el trabajo en varias columnas.

Por ejemplo, para imprimir el archivo `fichero.txt` con la cabecera “Cabecera”:

```
pr -h Cabecera fichero.txt | lpr
```

La orden **lpq** informa de los trabajos pendientes en la cola de impresión de la impresora por defecto, o de la impresora que se especifique con la opción `-P`, con el siguiente formato:

```
Orden Propietario Id del trabajo Nombre del fichero Tamaño
```

La orden **lprm** `<Id-trabajo>` permite cancelar un trabajo (aunque ya haya comenzado a imprimirse). También podemos especificar la cola de impresión de la impresora de la cual queremos que se elimine el trabajo con la opción `-P`.

La orden **lpc** informa acerca del estado del sistema de impresión y administrarlo. Por ejemplo, `/usr/sbin/lpc status` informa del estado de las colas de impresión de todas las impresoras. Con `lpc help` se obtiene información de todos los comandos válidos. Nótese que se debería deshabilitar una impresora cuando se vayan a hacer cambios en su configuración.

Por último, en casi todas las distribuciones de Linux es ya posible encontrar el sistema de impresión **CUPS** (*Common Unix Printing System*) que permite, mediante una sencilla interfaz *web*, la administración de impresoras y colas de impresión. Además, muchas distribuciones incorporan aplicaciones gráficas que interactúan con CUPS.

6. DISCOS

Sobre los dispositivos de bloques se suelen crear **sistemas de ficheros**. De cada sistema de ficheros que estemos utilizando nos interesan los siguientes datos:

- El espacio total para datos, el espacio ocupado y el espacio disponible. Estos datos se suelen expresar en unidades o bloques lógicos de 1 KB.
- El número de ficheros y directorios totales que se pueden crear, cuántos se han creado ya y cuántos más podemos crear. Estos datos se expresan en términos de nodos-i. El concepto de **nodo-i** sólo tiene sentido cuando nos referimos a los sistemas de ficheros nativos de Linux, es decir, `ext2` o `ext3`. El número de nodos-i libres indica el número total de ficheros y directorios que podemos crear todavía.

La orden **df** informa del espacio total, ocupado y disponible para cada uno de los sistemas de ficheros, con el siguiente formato:

S.ficheros	Bloques de 1K	Usado	Dispon	Uso %	Montado en
/dev/hda3	11820088	5320096	5899564	48 %	/
/dev/hda2	6822312	4840308	1982004	71 %	/dos

La opción `-i` sustituye los bloques por nodos-i, y la opción `-T` informa además del tipo de sistema de ficheros en cada caso.

La orden **du** `<directorio>` nos muestra cuántos bloques lógicos de 1 KB ocupa el directorio indicado y cada uno de sus subdirectorios. Si utilizamos la opción `-a`, también aparecen los ficheros, y si utilizamos la opción `-s` obtenemos un resumen sin que aparezcan los detalles. Téngase en cuenta que también se contabilizan los bloques lógicos ocupados por los propios directorios que, como hemos dicho, son ficheros especiales.

7. ÓRDENES Y PROGRAMAS ÚTILES

7.1. La orden **find**

La orden **find** es una orden muy potente que se utiliza para buscar ficheros y directorios que cumplan ciertos criterios. Además, permite especificar operaciones a ejecutar sobre cada uno de los ficheros y directorios encontrados. Su formato es el siguiente:

```
find <directorio> <expresiones>
```

find busca en el directorio (o directorios) indicado y en todos sus subdirectorios de forma recursiva. Algunos de los criterios que se pueden utilizar para realizar la búsqueda son:

- **-name** `<nombre>`: nombre del fichero a buscar (se pueden utilizar comodines, en cuyo caso, se debería encerrar el nombre entre comillas).
- **-iname** `<nombre>`: similar al anterior pero sin distinguir mayúsculas y minúsculas.
- **-user** `<usuario>`: propietario al que debe pertenecer el fichero.
- **-group** `<grupo>`: grupo al que debe pertenecer el fichero
- **-type** `c|b|d|l|f`: tipo del fichero (carácter, bloque, directorio, enlace simbólico o fichero regular).
- **-mtime** `[+|-]n`: fichero modificado hace más de, menos de o exactamente n días.
- **-atime** `[+|-]n`: fichero utilizado hace más de, menos de o exactamente n días.
- **-size** `[+|-]n`: fichero con un tamaño de más, menos o exactamente 512 x n bytes.

- **!**: negación de un criterio.
- **-o**: O lógico de criterios.
- **-a**: Y lógico de criterios (esta es la operación lógica implícita cuando se indican varios criterios).

Podemos agrupar los criterios con paréntesis y combinarlos mediante operaciones lógicas. La única precaución a tomar es que los paréntesis se especifican como “\ (“ y “\)” (los paréntesis en la línea de órdenes tienen un significado especial que se verá en la segunda práctica, por eso deben ir precedidos de un carácter “\”, para “quitarles” ese significado especial).

Algunos ejemplos de uso de esta orden:

- `find /usr -type f -size +1k -o !-user miguel -print`: busca, a partir del directorio `/usr`, aquellos ficheros regulares cuyo tamaño sea superior a 1 KB o no pertenezcan al usuario miguel, y muestra los nombres de aquellos ficheros que cumplan el criterio.
- `find . -group proyecto -exec chmod a+rw {} \;`: busca, a partir del directorio actual, todos los ficheros y directorios que pertenezcan al grupo proyecto, y sobre cada uno de los que encuentra ejecuta la orden especificada con `-exec`. Nótese el espacio entre “}” y “\”, y que la expresión “{ }” será sustituida por el nombre de cada uno de los ficheros que se encuentren.

7.2. La orden tar

La orden **tar** nos permite archivar y comprimir varios ficheros y directorios en un único fichero. Por ejemplo:

- `tar cvzf resultado.tar.gz directorio1 fichero1 fichero2 directorio2`: empaqueta y comprime el contenido de los directorios `directorio1` y `directorio2`, y de los ficheros `fichero1` y `fichero2` dentro del fichero `resultado.tar.gz`.
- `tar tvzf resultado.tar.gz`: lista el contenido del fichero previamente creado `resultado.tar.gz`.
- `tar xvzf resultado.tar.gz`: descomprime y desempaqueta el contenido del fichero previamente creado `resultado.tar.gz` en el directorio actual.

Si se omite la opción `f`, la salida será la salida estándar y la entrada será la entrada estándar, y el nombre del fichero no es necesario. Si se omite la opción `z`, no se realiza la compresión/descompresión, y la extensión de los ficheros es `.tar` en lugar de `.tar.gz`.

7.3. Otras órdenes y programas de uso común

Este apartado presenta una listado de otras órdenes y programas que se utilizan habitualmente desde la línea de órdenes:

- **date** muestra la fecha y hora actuales, y permite modificarlas, mientras que **cal** muestra un calendario del mes actual o del que se especifique como parámetro.
- **more** y **less**: permiten visualizar un fichero y realizar búsquedas en él.
- **clear** limpia la pantalla y **reset** inicializa el terminal asociado a la sesión actual. Lo último es necesario cuando se “corrompe” el mapa de caracteres del terminal. Esto puede ocurrir, por ejemplo, cuando al ejecutar `cat fichero.bin`, siendo `fichero.bin` un fichero binario.
- **attrib**, **mcd**, **mcopy**, **mdel**, **mdir**, **mformat**, **mlabel**, **mmd**, **mrd**, **mren** y **mtype** (`mttools` permiten trabajar con un disquete de Windows o con una partición del disco duro del tipo FAT16, sin necesidad de montarlos. Estas herramientas reconocen las unidades `A:`, `B:`, `C:`, etc., pero las rutas de ficheros y directorios se deben especificar con el separador de Linux “/”, por ejemplo, `mdir a:/directorio`.

- **uname -a**: muestra información sobre el tipo de CPU, el nombre de nuestra máquina, el sistema operativo, la versión del kernel, etc.
- Por último, las órdenes `grep`, `sort`, `cut`, `paste`, `join`, `tr`, `wc`, `nl`, `head` y `tail` (a veces conocidas con el sobrenombre de *filtros*), entre otras, realizan operaciones de manipulación de diversa naturaleza sobre ficheros de texto y, como veremos, se utilizan con frecuencia en la construcción de guiones `shell`. En el web de la asignatura se puede encontrar un documento con numerosos ejemplos ilustrativos de usos comunes de algunas de estas órdenes, y en sus respectivas páginas del manual sus detalles de funcionamiento.

7.4. Conexión remota y transferencia de ficheros: telnet, ftp, y ssh

El programa `telnet host` permite abrir una sesión en un ordenador remoto, en el que dispongamos de una cuenta, y trabajar como si estuviésemos sentados delante de él. Al finalizar la sesión, con `exit`, `logout`, o pulsando **CTRL+D**, también terminaría la ejecución del programa `telnet`.

Por su parte, el programa orden `ftp host` permite transferir ficheros desde/hacia un ordenador remoto, en el que dispongamos de una cuenta. Con `ftp help` obtenemos una lista de los comandos disponibles.

Los programas `telnet` y `ftp` se usan cada vez menos. Por un lado, `telnet` ha sido desplazado por el programa `ssh` debido a los problemas de seguridad que supone transmitir el usuario y su contraseña como texto plano hasta el ordenador remoto. Por otro lado, el programa `ftp` además de plantear los mismos problemas de seguridad que el programa `telnet`, ha sido reemplazado por herramientas gráficas, como `gFTP`, e incluso por navegadores que soportan el protocolo FTP.

Los programas `ssh` y `scp` desempeñan el mismo papel que `telnet` y `ftp` pero con ellos la transmisión tanto del usuario y la contraseña como de los datos se realiza de forma cifrada lo que garantiza la privacidad de las comunicaciones.

Algunos ejemplos de uso de estos programas:

- `ssh antonio@host.com`: conexión del usuario `antonio` al ordenador remoto `host.com`.
- `scp flocal pedro@host.com::` transfiere el fichero local `flocal` a la cuenta del usuario `pedro` en el ordenador remoto `host.com`.
- `scp pedro@host.com:fremoto .:` transfiere el fichero remoto `fremoto` desde la cuenta del usuario `pedro` en el ordenador remoto `host.com` hasta el directorio actual.

8. X-WINDOW

8.1. Breve historia de X-Window

X-Window System nace en la década de los 80 en el Instituto Tecnológico de Massachusetts (MIT). En los almacenes de dicha institución se acumulaban decenas de estaciones de trabajo de distintas plataformas, procedentes de donaciones y adquisiciones por parte del MIT. Se pensó en utilizar dichas máquinas con fines educativos, pero resultaba imposible debido a que las máquinas tenían muy diversas arquitecturas y, por tanto, ejecutan distintos sistemas operativos. No obstante, un equipo de trabajo del mismo instituto encontró la solución: crear un entorno de ventanas de red. Este sistema debía ser independiente de la plataforma y debía poder ejecutar aplicaciones tanto locales como remotas.

Con esta finalidad se crea el proyecto Athena, con la colaboración de dos gigantes de la informática como DEC e IBM. Un año después surge X-Window System como el primer entorno de ventanas de red completamente independiente del hardware.

Hoy día X-Window System sigue en manos del MIT. Aunque éste lo distribuye gratuitamente, se reserva los derechos de autor sobre él. Ha sido portado a gran cantidad de plataformas y sistemas operativos pero es en el mundo de UNIX donde ha tenido la mayor aceptación.

8.2. X-Window System

X-Window es un sistema cliente/servidor dónde el cliente y el servidor pueden estar ejecutándose en máquinas diferentes o, como es común en la mayoría de ordenadores personales, pueden residir en la misma máquina.

En X-Window la relación cliente/servidor es aparentemente opuesta a lo que se entiende habitualmente. En X-Window el servidor provee servicios para poder usar la pantalla, el teclado y el ratón, mientras que los clientes son las aplicaciones que utilizan estos recursos para interactuar con el usuario. De este modo mientras el servidor se ejecuta de manera local, las aplicaciones pueden ejecutarse remotamente en otras máquinas, proporcionando así lo que se conoce como **transparencia de red**.

De entre todos los clientes, hay uno especial que es el que determina la apariencia del escritorio y de las ventanas, y que se denomina **gestor de ventanas** o *window manager*. Existen varios como: Enlightenment, Fluxbox, Sawfish, Fvwm, Fvwm95, mwm, olvwm, ctwm, etc. También hay otros que no son “simples” gestores de ventanas sino verdaderos entornos integrados. Entre ellos, KDE y GNOME se han convertido en el estándar *de facto* pues son incluidos por la mayor parte de las distribuciones de Linux.

No describiremos el modo de funcionamiento de ningún gestor de ventanas en particular, pero sí vamos a mencionar algunos mecanismos que podemos utilizar en todos ellos. El servidor se ejecuta con **startx** y se sitúa en el terminal virtual nº7. Una vez dentro del entorno de ventanas de X-Window, con CTRL+ALT+Fx, donde donde Fx es F1,2,3,4,5,6 podemos situarnos en cualquiera de los 6 terminales virtuales que existen en modo texto. Con ALT+F7 (o CTRL+ALT+F7), regresamos a X-Window. Es posible ejecutar más de una copia del servidor. Por ejemplo, si ejecutamos `startx -- :0&`, volvemos al terminal de texto desde donde ejecutamos lo anterior y volvemos a ejecutar `startx -- :1&`, habremos conseguido dos instancias de X-Window, una en el terminal virtual 7 y otra en el 8.

Una aplicación basada en X-Window se puede ejecutar de dos formas. Si se encuentra en alguno de los menús que proporciona el gestor de ventanas, seleccionándola de dicho menú. Si tenemos abierto un terminal, ejecutándola directamente desde dicho terminal (en este caso interesa ejecutar el programa en segundo plano mediante el carácter ‘&’ para que así podamos seguir utilizando dicho terminal). Muchos de los programas para X-Window se suelen encontrar en el directorio `/usr/X11R6/bin`.

Ya hemos dicho que X-Window se basa en la filosofía cliente/servidor. Cuando ejecutamos `startx` estamos iniciando el servidor. Los clientes son los distintos programas que ejecutamos dentro del entorno X-Window como son `xterm` o `xclock`. Por tanto, podemos ejecutar el servidor en nuestro ordenador local y ejecutar los clientes en un ordenador remoto. Para ello los pasos a seguir son:

1. Ejecutar `startx` en la máquina local (si no se está ejecutando ya un servidor gráfico).
2. Abrir una sesión en la máquina remota con `ssh`, con la opción `-X`.
3. Ejecutar la aplicación X-Window (cliente) en el ordenador remoto, por ejemplo, `xclock`.

8.3. Aplicaciones para X-Window System

Existen muchísimas aplicaciones para X-Window. Entre otras muchas podríamos destacar las siguientes:

- StarOffice/OpenOffice: Se trata de una *suite* de ofimática que incluye un procesador de textos, editor de ecuaciones, hoja de cálculo, etc.
- Mozilla Firefox/Thunderbird: navegador web y cliente de correo.
- Acroread: visualizador de documentos en PDF.
- GIMP: aplicación de retoque fotográfico de apariencia y funcionamiento similar a Photoshop.
- XMMS: reproductor de CDs y de ficheros MP3, OGG, WAV, etc.
- K3b: aplicación para grabar CDs y DVDs.
- Gaim: cliente de mensajería instantánea que admite los protocolos MSN, Yahoo, IRC, etc.
- Ekiga: aplicación de telefonía IP.

9. EJERCICIOS

1. Crea un alias de la orden `mv`, también llamado `mv`, para que al utilizar dicho alias se muestren los ficheros que se mueven y, en caso de que exista el destino, se nos pida confirmación de sobrescritura. (*Pista:* ejecuta `help alias`).
2. Utiliza las opciones adecuadas tanto en `ls` como en `less` para que al utilizar estas dos órdenes en una tubería como `ls | less` podamos ver los ficheros por columnas y coloreados. Esto es útil para ver el contenido de directorios en los que existe un gran número de ficheros, como ocurre con `/usr/bin`.
3. Utiliza la orden `split` para dividir un fichero de nombre `administracion`, que tiene un tamaño de 15 MB, en fragmentos con un tamaño no superior a 1400 KB (para que quepan en disquetes). Todos los ficheros deben tener un nombre que empiece por `admin`. ¿Cómo podemos unir todos los fragmentos para obtener de nuevo el fichero original?
4. Establece los permisos adecuados para que el fichero `~/temp/solucion.txt` sólo pueda ser borrado por el propietario, leído y modificado por el propietario y el grupo, y leído por el resto de usuarios.
5. Supongamos que tenemos el fichero de texto `~/trabajos/prac.tex`. Modificar los permisos del directorio y del fichero para que el dueño pueda hacer cualquier cosa con el fichero (borrarlo, modificarlo, leerlo, etc.), los de su grupo puedan leerlo y modificarlo, pero no borrarlo, y el resto de usuarios no tenga ningún acceso sobre él (ni siquiera ver sus atributos).
6. Averigua por qué podemos crear y borrar ficheros y directorios en el directorio `/tmp` pero, sin embargo, no podemos borrar los ficheros y directorios que no hemos creado nosotros. (*Pista:* Buscar información sobre el llamado *sticky bit*).
7. Muestra todos tus procesos y los del resto de usuarios que tengan asociada una terminal.
8. Un directorio que pertenece al usuario `juan` y al grupo `users`, tiene los permisos `rwxr-x--`. Dentro de ese directorio existen únicamente dos ficheros, `fich1` y `fich2`, también pertenecientes al usuario `juan` y al grupo `users`, y con permisos `rw-rw-r-` y `r-xr-xr-x`, respectivamente. Supongamos también que existe otro usuario llamado `pedro` que también pertenece al grupo `users`. Determina si las siguientes afirmaciones son verdaderas o falsas:
 - `juan` puede modificar `fich1`
 - `pedro` puede borrar `fich1`
 - cualquier usuario puede ejecutar `fich2`
 - `juan` puede cambiar los permisos de `fich2`
 - el superusuario puede modificar `fich1`
9. Imprimir en la impresora “HP5L” el contenido del directorio `/usr/lib` mostrando, para cada fichero, su nombre y su número de nodo-`i`. Los ficheros deben estar ordenados por número de nodo-`i`, y las líneas deben aparecer numeradas. (*Pista:* usar comandos `sort` y `nl`).
10. En un directorio hay un conjunto de ficheros `.dbf` que corresponden a las bases de datos de una aplicación. Para analizarlos mejor se quiere producir una lista alfabética con todos los ficheros numerados a la izquierda (con 5 dígitos y ceros a la izquierda) de la siguiente manera:

```
00001  albaranes.dbf
00002  aplicaciones.dbf
...    ...
```

(*Pista:* buscar opciones `nl` en la documentación).

11. Utilizando las órdenes `df` y `df -i` hemos obtenido la siguiente información:

Filesystem	1k-blocks	Used	Available	Use%	Mounted on
/dev/hda3	51916	36044	13191	73%	/

Filesystem	Inodes	IUsed	Ifree	%IUsed	Mounted on
/dev/hda3	13440	6587	6853	49%	/

En esta condiciones, ¿cuántos directorios vacíos podemos crear? ¿Y cuántos ficheros con un tamaño de 2300 bytes?

12. Guarda en un fichero una copia del Master Boot Record (MBR) del disco duro de tu ordenador si suponemos que el MBR es el primer sector del disco y que los sectores del disco son de 512 bytes. (*Pista*: usa `dd`).
13. En un ordenador con una única disquetera, indica cómo realizarías una copia idéntica de un disquete. ¿Y si el ordenador dispone de dos disqueteras idénticas?
14. Comprime con la orden `tar` todos los ficheros de tu directorio de usuario y de sus subdirectorios. El fichero se llamará `trabajo.tar.gz`.
15. Utiliza la orden `tar` para crear un archivo con todos los ficheros que se encuentran en nuestro directorio de trabajo, cuyos nombres terminen en `.c.o.h`, y que se hayan modificado con posterioridad al 5 de octubre de 2006.
16. Cambia todos los ficheros terminados en `.c` y `.h` para que terminen en `.c.orig` y `.h.orig`, respectivamente. (*Pista*: usar la orden `find`).
17. Encontrar todos los ficheros que no puedan ser leídos por su propietario (*Pista*: investigar el uso de la opción `-perm` de la orden `find`).
18. Encontrar todos los ficheros que no puedan ser borrados por el propietario del directorio en el que residen.
19. Buscar y comprimir con `gzip` (utilizando compresión máxima) los ficheros del usuario actual que no se hayan modificado en 90 días.
20. Buscar todos los ficheros regulares del propietario `bin` que existan a partir del directorio `/usr`, que se hayan modificado hace más de 3 días y que tengan un tamaño superior a 1 KB.
21. Buscar todos los ficheros regulares del propietario `bin` en el directorio `/usr` y que se hayan modificado hace menos de una semana y que tengan un tamaño inferior a 5KB.
22. Imprimir todos los ficheros con `tags` de CVS, esto es, los ficheros del directorio actual y de sus subdirectorios que contienen la identificación “`$Id: ...`”.
23. Imprime una lista de todos los ficheros regulares modificados en los últimos 30 días y que, o bien pertenecen al `root` y tienen un tamaño inferior a 10KB, o bien pertenecen a cualquier otro usuario. La búsqueda se debe hacer en los directorios `/usr` y `/home`.
24. Calcular la máxima profundidad del subárbol personal (directorio de entrada). (*Pista*: usar la orden `find` con la opción `-printf`, combinada con `sort` y `tail`).
25. Utilizando una única línea órdenes (y sin usar el separador `;`) guarda en un fichero llamado `directorios` los nombres (uno por línea) de todos los subdirectorios existentes en el directorio `/home` (sin incluir los subdirectorios de los subdirectorios), e imprime también en pantalla un listado a tres columnas balanceadas y con las filas de las columnas numeradas. (*Pista*: usar las órdenes `tee` y `pr`).

26. Imprime en la impresora LJ4 y guarda en el fichero `ultimoscambios` una lista de todos tus ficheros regulares modificados en los últimos 5 días. Las líneas para cada fichero deben tener el siguiente formato:

```
[<nombre fichero>:<tamaño>]-><día-mes-año de última modificación>
```

Por ejemplo, una línea podría ser:

```
[trabajo.txt:5134]->12-10-2002
```

27. Utilizando tuberías y en una única línea de órdenes, mostrar todos los usuarios que estén ejecutando el programa `bash`. Es posible que un mismo usuario esté ejecutando el programa varias veces, aunque en la salida final sólo deberá aparecer una vez. (*Pista*: Éste y los que siguen son típicos ejercicios que combinan mediante tuberías varios comandos de los denominados *filtros*, como `tr`, `cut`, `grep`, `uniq`, `sort`, etc.).
28. Mostrar sólo el nombre de los dispositivos que estén a una carga de uso del 90 % o más. (*Pista*: usar opción `-E` del `grep`).
29. Muestra los procesos cuyo propietario es `root` ordenados por porcentaje de uso de CPU.
30. Mostrar el proceso que está utilizando más memoria y el porcentaje que usa (*Pista*: Las opciones del `sort` pueden facilitarnos el trabajo).