Tema 1

El Entorno de Trabajo

En este capítulo se introducen las bases sobre el entorno de trabajo para la programación en C++. El tema comienza con una visión general de los sistemas operativos y una descripción del sistema operativo Linux, sobre el que se desarrollarán las prácticas. A continuación se presenta una descripción de la interfaz gráfica de usuario KDE, sus principales características y utilidades. Las herramientas utilizadas para el desarrollo de programas en C++ bajo Linux serán: la *shell bash*, el editor *GVim*, el compilador GCC y el depurador GDB/DDD. De cada una de estas herramientas se describen en este tema sus características básicas, dando la suficiente información para el desarrollo de las prácticas en el laboratorio a lo largo del cuatrimestre.

Índice General

1 El Entorno de Trabajo **1**

1.1 Sistemas operativos 3

1.1.1 El sistema operativo Linux 4

1.1.2 Terminales e interfaces gráficas de usuario 6

1.2 Herramientas de programación en C++ bajo Linux 12

1.2.1 El editor GVim 12

1.2.2 El compilador GCC 15

1.2.3 El depurador de código GDB/DDD 18

1.3 Algunos enlaces importantes 19

1.4 Notas sobre la distribución Linux/Knoppix 21

1.5 El compilador GCC sobre Windows 24

1.1 Sistemas operativos

Un sistema computador consta de distintos recursos hardware y software. El propósito principal de un sistema operativo es facilitar la utilización sencilla, eficiente, equitativa, ordenada y segura de estos recursos. Permite a los usuarios del computador emplear *software de aplicación*: hojas de cálculo, procesadores de texto, navegadores web, software de correo electrónico, etc. Los servicios principales que ofrece un sistema operativo son:

- la ejecución de programas,
- las operaciones de entrada y salida realizadas por los programas,
- la comunicación entre procesos,
- la detección y notificación de errores y
- la manipulación de archivos de todo tipo.

Los sistemas operativos pueden ser de diferente tipo:

- Sistemas de un solo usuario y un solo proceso: MacOs, DOS, Windows 3.1, etc.
- Sistemas de un solo usuario y múltiples procesos: OS/2, Windows 95/98, Windows NT Workstation, etc.
- Sistemas de múltiples procesos y múltiples usuarios: UNIX, LINUX, Windows NT Server, etc.

1.1.1 El sistema operativo Linux

Estrictamente hablando, Linux es el *kernel* (núcleo) del sistema operativo de una computadora. Un *kernel* es el software que permite la comunicación entre las aplicaciones del ordenador y el hardware, suministrando servicios como la gestión de ficheros, memoria virtual, dispositivos de entrada/salida, etc. El *kernel* de Linux fue creado por Linus Torvalds y se anunció por primera vez en la Internet en 1991.

La organización GNU (*Gnu is Not Unix*) escribió y desarrolló muchas de las aplicaciones de software que, combinadas con el *kernel* de Linux, constituyen un 'sistema operativo completo'. Entre estas aplicaciones se encuentran una serie de compiladores, editores y herramientas de desarrollo de software bajo la denominada *licencia pública general* (*General Public License*, o GPL).

Todo este software, junto con algunas utilidades para instalación y configuración, constituyen lo que se denomina una *distribución* de Linux. Cada distribución está creada por una persona, conjunto de personas, o bien una empresa. En este momento, hay más de 250 distribuciones de Linux, que se utilizan en un amplio espectro de plataformas hardware, que van desde grandes arquitecturas de procesadores de 64 bits y servidores de Internet, hasta diminutos procesadores empotrados, e incluso relojes de pulsera. Las principales distribuciones de Linux son: RedHat, Debian, Mandrake, Caldera, Corel, Slackware, SuSE y Knoppix.

En las prácticas de laboratorio se usará la distribución RedHat (versión 9.0), que es una de las más difundidas. La distribución está 'instalada' en el disco duro de los ordenadores del laboratorio, conviviendo con el sistema operativo Windows. Esto se consigue mediante la partición del disco duro del ordenador en diferentes secciones: al menos una por cada sistema operativo instalado. Cuando el ordenador del laboratorio arranca (se inicia su funcionamiento), el usuario tiene la posibilidad de seleccionar entre los diferentes sistemas operativos instalados.

Para trabajar en casa, el alumno puede descargarse de la Internet cualquiera de las diferentes distribuciones que se ofrecen gratuitamente (véase sección 1.3). La distribución aconsejada para el uso en casa es la Knoppix 3.6. Knoppix es un CD 'arrancable' con una colección de programas GNU/Linux. Posee herramientas para la detección automática de hardware y soporta multitud de tarjetas gráficas, tarjetas de sonido, dispositivos SCSI y USB y otros periféricos. No es necesario instalar nada en el disco duro de nuestro ordenador para usarlo, por lo que se aconseja en aquellos casos en los que no se quieran modificar las particiones y/o sistemas operativos existentes en el ordenador. Todas las herramientas de desarrollo en C++ están incluidas en el CD de la distribución Knoppix 3.6.

Características y uso del sistema operativo Linux

Como se ha comentado en la sección 1.1, el sistema operativo Linux es un sistema multiusuario y multiproceso. Esto quiere decir que, simultáneamente, puede haber múltiples usuarios utilizando el sistema y, a su vez, estos usuarios pueden estar ejecutando múltiples aplicaciones al mismo tiempo. Es el sistema operativo el que se encarga de gestionar toda esta *multitarea* para que la ejecución de los procesos tenga la apariencia de estar ocurriendo concurrentemente para todos los usuarios. Es más, un mismo usuario puede abrir simultáneamente múltiples sesiones en el sistema. Esto se consigue porque hay diferentes maneras de acceder al sistema: por un lado, mediante diferentes terminales de texto, gráficas, *shells*, etc., que se usan directamente sobre el ordenador; y, por otro lado, mediante accesos remotos (por la red) al ordenador desde diferentes terminales.

Cada uno de los usuarios que pueden usar el sistema deben tener una *cuenta* abierta en ese sistema. Ello les proporcionará un *nombre de usuario* y una *contraseña* para el acceso *autentificado* al sistema, además de algunos directorios personales y exclusivos para trabajar (el directorio `home` del usuario). Existe un usuario especial, denominado *administrador* o *root*, que se encarga de la administración del sistema y de la gestión de las cuentas de los usuarios.

Linux maneja toda la información de los discos (disco duro, *floppy*, etc.) mediante ficheros. Todos los datos que se guardan en el disco deben pertenecer a algún fichero. Incluso, todos y cada uno de los dispositivos periféricos del ordenador son manejados por Linux mediante ficheros. Un fichero se identifica, entre otras cosas, por su nombre. Además, los ficheros están repartidos por una estructura jerárquica de directorios que permite organizarlos eficientemente: los ficheros del sistema por un lado, por otro los ficheros de usuarios, etc. Esa estructura jerárquica se denomina también *árbol de directorios*, cuya raíz se identifica con una barra de dividir: /. De ese directorio raíz 'cuelgan'

una serie de directorios importantes en el sistema, como son:

- `/home`: directorio de los usuarios del sistema.
- `/root`: directorio del administrador del sistema.
- `/etc`: directorio de configuración del sistema.
- `/usr`: contiene, entre otras cosas, librerías de programas y páginas de manuales.
- `/mnt`: es usado por los administradores del sistema para montar otros sistemas de ficheros.
- `/bin`: contiene la mayoría de los ejecutables del sistema.
- `/boot`: contiene todos los archivos necesarios para el arranque de Linux.
- `/var`: datos de tamaño variable del sistema.
- `/dev`: ficheros de dispositivos del sistema.

Estos directorios pueden estar en la misma o diferentes particiones del disco duro. Además, cada uno de estos directorios se ramifica en su propio árbol de subdirectorios.

Cada uno de los ficheros de un sistema de archivos de Linux *pertenece* a un usuario concreto y/o grupo de usuarios. Además de su nombre y la ruta (directorio) donde está guardado, cada archivo está caracterizado por una serie de atributos que identifican, por un lado, el tipo de archivo (directorio, vínculo, dispositivo, etc.) y, por otro, los *privilegios* de acceso que los diferentes usuarios tiene sobre el archivo. Un archivo en Linux puede ser leído, escrito o ejecutado por su propietario, por otros usuarios del grupo al que pertenece el propietario o por el resto de los usuarios del sistema. Para cada uno de estos subgrupos de usuarios se establecen privilegios que restringen o permiten las operaciones de lectura, escritura y ejecución del archivo. De esta forma, se puede proteger eficazmente el acceso y la seguridad de que cada uno de los archivos del sistema. El administrador del sistema tiene privilegios para leer, escribir, modificar o borrar todos los archivos del sistema, sea cual sea su propietario.

1.1.2 Terminales e interfaces gráficas de usuario

Como se ha comentado en el apartado 1.1.1, el sistema operativo Linux es, en realidad, un *kernel* que permite al usuario y sus aplicaciones comunicarse con el hardware del ordenador. El usuario puede gestionar los distintos componentes hardware usando como

interfaz el sistema operativo. Para ello, es necesario que el usuario disponga de algún medio de comunicación para 'transmitirle' al sistema operativo sus peticiones u 'órdenes'.

La manera más elemental de hacer esto es mediante el uso de comandos de texto. Por ejemplo, si el usuario introduce la expresión `ls -al` en el lugar adecuado, debería obtener como respuesta, visualizada en la pantalla de su ordenador, la lista de los ficheros contenidos en el directorio actual de trabajo del disco. Ese *lenguaje de comandos* para 'hablarle' al sistema operativo implica, por un lado, la necesidad de que el usuario aprenda un conjunto más o menos extenso de órdenes, opciones, parámetros, etc; y, en segundo, lugar, la necesidad de que exista un *intérprete de comandos* para traducir esos comandos al lenguaje que el sistema operativo entiende. Esos intérpretes de comandos se denominan *shells*.

El sistema operativo Linux ofrece interfaces textuales para la introducción de comandos y visualización de resultados. Una vez que el sistema operativo se ha cargado en la memoria del ordenador, en la pantalla aparecerá un mensaje de texto con la versión del sistema operativo y la palabra *login*. Es necesario introducir un nombre de usuario válido en ese ordenador y, posteriormente, una contraseña para 'autentificarse' y permitir así el uso del sistema. Una vez que se ha introducido un *login* y una clave correctas, se accede directamente al intérprete de comandos para poder trabajar con el sistema.

Además, la mayoría de las distribuciones de Linux facilitan la interacción entre el usuario y el sistema operativo mediante la oferta de una o varias *interfaces gráficas de usuario* (GUI), también llamadas *administradores de escritorio*. Estas GUI permiten al usuario interactuar con el sistema mediante el uso de iconos, botones, barras de herramientas, menús, ventanas, *clicks* de ratón, etc., por lo que es más 'amigable' el uso del ordenador. Las GUI de Linux son *independientes* del *kernel* del sistema operativo. Son, simplemente, otras aplicaciones que se ejecutan en el sistema. La interfaz entre una GUI y el sistema operativo lo constituye el denominado *servidor XFree86*. Si el sistema operativo está debidamente configurado, la GUI se ejecutará automáticamente una vez que el sistema operativo está cargado totalmente. De esta manera, se consigue que desde el principio el usuario pueda trabajar en 'modo gráfico' y prescindir, si lo desea, de la comunicación mediante *shells*. Si la GUI no se ha cargado automáticamente, se puede iniciar con el comando `startx`.

Entre los GUI de Linux más utilizados destacan GNOME y KDE. Ambos poseen numerosas utilidades, aplicaciones, herramientas de administración y configuración, etc., integradas en un escritorio.

El administrador de escritorio KDE

El *K Desktop Environment* (KDE, entorno de escritorio K) fue desarrollado por una organización formada por programadores voluntarios. KDE es un *sistema integrado*, porque proporciona una implementación coherente y uniforme de funciones, como la interfaz de programación de aplicaciones (API), administración de ventanas, herramientas de configuración de escritorio, administrador de sesiones y, sobre todo, programas de aplicación. Antes de acceder al escritorio de KDE es necesario autentificarse como usuario, mediante el correspondiente nombre y clave introducida en la ventana de presentación gráfica del sistema.

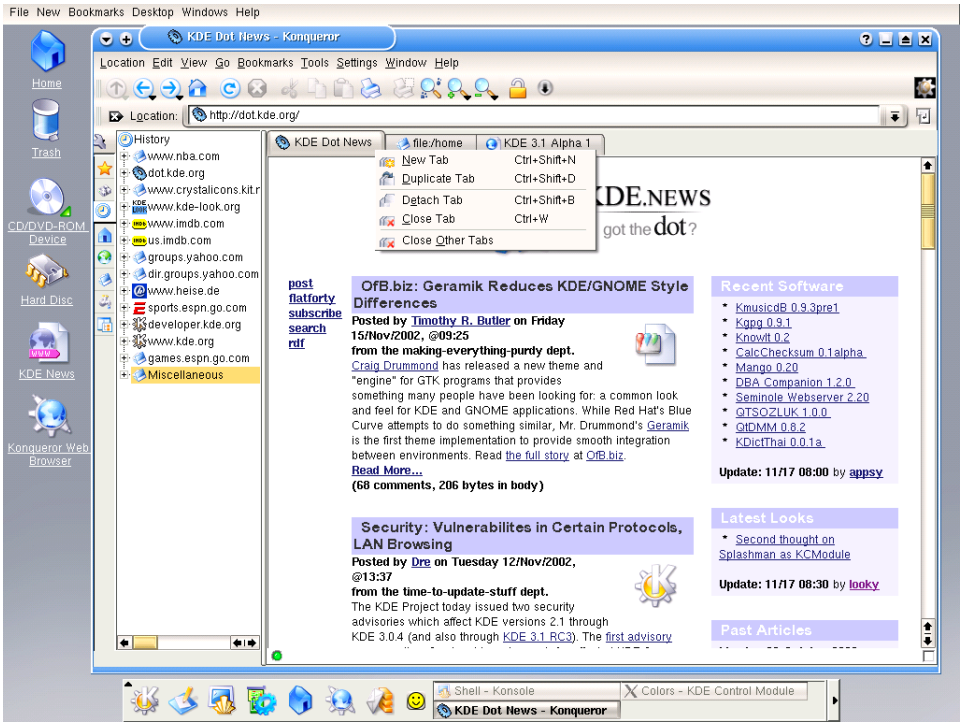

Figura 1.1: Administrador de escritorio KDE

El escritorio de KDE tiene un aspecto muy similar al de otros sistemas de escritorio como Macintosh y Windows 98/2000/XP (véase figura 1.1). Una de las diferencias más notables entre el funcionamiento de KDE y de Windows o Macintosh es que para lanzar un programa en KDE se utiliza un único *click* de ratón, en lugar de dos (aunque esto es configurable). En la figura 1.1 puede observarse una de las principales utilidades de KDE, la *barra de herramientas*, situada en la parte inferior de la figura. Esta barra o panel actúa como centro de información y lanzadera para muchas de las capacidades y programas de aplicación del escritorio: las *shells* de comandos, editores, navegadores web, panel de control (configuración del sistema), etc. El *botón K* permite el acceso al

menú de KDE, desde donde se puede acceder a la mayoría de las aplicaciones instaladas en el ordenador.

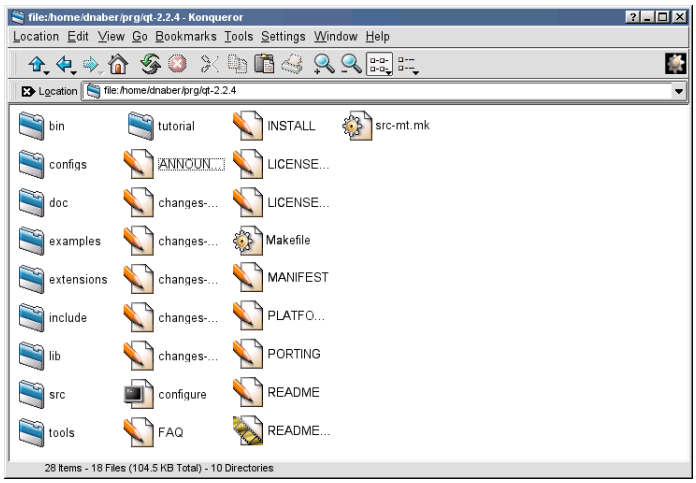

Figura 1.2: Navegador Konqueror

Una de las principales utilidades de KDE es el *navegador Konqueror* (véase figura 1.2). Esta herramienta permite tanto la navegación por Internet como la gestión de los directorios y archivos del sistema. En la barra de localización (*Location* en la figura 1.2) aparecerá la URL correspondiente (`http:` o `file:`). Como gestor de archivos y directorios funciona de manera similar al *Explorador de Windows*, con la excepción de que el acceso a los ficheros y directorios se hace mediante un solo *click* de ratón. Mediante los iconos, el menú o el botón derecho del ratón se puede copiar, pegar, mover, eliminar, etc., ficheros y directorios (siempre que tengamos los 'permisos' necesarios para hacer estas operaciones sobre esos elementos). La ventana de *Konqueror* se puede dividir en varias ventanas para facilitar las operaciones de copia y movimiento de ficheros. Cada fichero o directorio se puede representar en la ventana de *Konqueror* en muchos formatos: lista, iconos grandes, pequeños, etc.

El intérprete de comandos

Como se ha comentado en el apartado 1.1.2, los intérpretes de comandos o *shells* permiten interactuar con el sistema mediante órdenes o comandos escritos con una sintaxis específica. Existen diversas *shells* en Linux, con coincidencias y diferencias entre ellas. Las más conocidos son: *bash*, *TC* y *Z*. En este apartado vamos a introducir algunas nociones sobre la *shell bash*.

El acceso a la *shell* puede hacerse a través de los terminales de texto (`tty`) de los que dispone el sistema (son los únicos de los que disponemos si no tenemos en-

torno de ventanas, o bien podemos acceder a ellos mediante la combinación de teclas `Control-Alt-F1-6`), o bien desde KDE (haciendo *click* en el icono correspondiente — normalmente representado con una pantalla y una concha— o bien, desde el menú K, seleccionando la opción correspondiente: `Menú K->Herramientas del Sistema->Terminal`).

Desde la *shell* puede realizarse, mediante comandos, todas las operaciones disponibles en el sistema. Se puede ejecutar una aplicación, simplemente indicando la ruta de un archivo ejecutable (que en Linux no tiene por qué tener extensión .EXE, sino únicamente permisos de ejecución). Así mismo, pueden administrarse los archivos del sistema: copiar, mover, eliminar, etc. Desde la *shell* también es posible realizar operaciones de configuración y actualización del sistema.

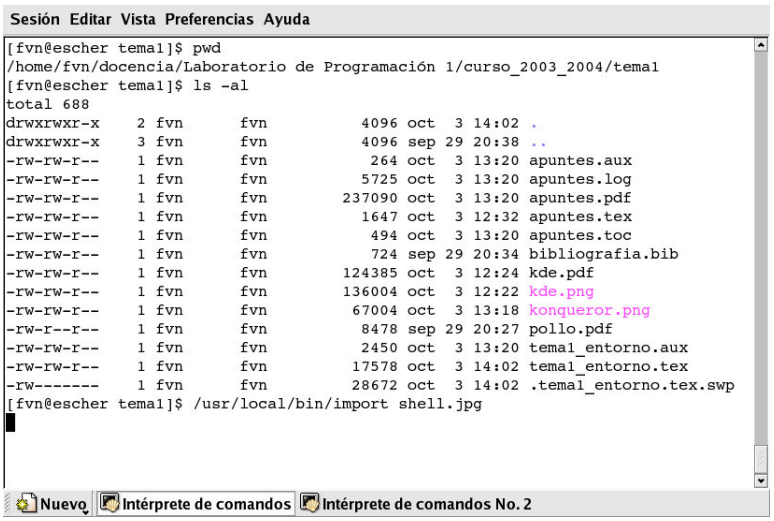

Figura 1.3: La shell bash

Cada línea de introducción de comandos está automáticamente encabezada por una expresión entre corchetes que indica el nombre de usuario, el nombre del ordenador (*hostname*) y la carpeta actual. Esa línea se denomina *prompt* (véase figura 1.3).

Algunos comandos de la shell bash

Los ficheros ejecutables (con permiso de ejecución), también llamados *binarios*, permiten lanzar aplicaciones, es decir, ejecutar programas, invocando su nombre. La mayoría de las veces es necesario indicar la ruta donde se haya el fichero binario, como ocurre en el último comando introducido en la figura 1.3, que lanza la aplicación `import`, situada en la ruta `/usr/local/bin/`, pasándole como parámetro la cadena `shell.jpg`. Sin embargo, existe una variable de entorno en el sistema que almacena en todo momento las posibles rutas donde 'buscar' los ficheros ejecutables. Esta variable, llamada `PATH`, se puede visualizar

mediante el comando `echo $PATH`. Si invocamos a un programa sin indicar la ruta, se buscará en cada uno de los directorios indicados en `PATH` para intentar localizar el archivo y lanzar la aplicación correspondiente.

Los comandos de la *shell bash* suelen tener numerosas opciones posibles que deben introducirse a continuación de su nombre. También pueden recibir parámetros. Algunos comandos de la *shell bash* son:

- `man <comando>`: visualiza la ayuda disponible en el sistema sobre algún comando. Por ejemplo, el comando `man ls` nos muestra ayuda sobre el comando `ls` (incluidas todas sus opciones de utilización y parámetros). Para salir de la ayuda basta con pulsar la tecla `q`.

- `pwd`: indica la ruta actual o directorio de trabajo (véase primer comando de la figura 1.3).

- `ls`: lista los ficheros contenidos en el directorio de trabajo. Opciones:

  - `-C`: muestra los ficheros en columnas.
  - `-a`: muestra los ficheros ocultos (aquéllos cuyo nombre empieza por punto).
  - `-l`: muestra los atributos de los ficheros. En la figura 1.3, el comando `ls -al` permite visualizar los ficheros, con sus atributos, incluyendo los ficheros ocultos. Véase como para cada fichero del directorio de trabajo se indica el tipo de fichero (`d` significa directorio y `-` fichero normal); los permisos de lectura (`r`), escritura (`w`) y ejecución (`x`) para el usuario propietario, su grupo y el resto de los usuarios, respectivamente; número de vínculos; propietario; grupo; tamaño en bytes; fecha y hora de su última actualización; y su nombre. El directorio de trabajo se representa mediante un punto (`.`), y su directorio padre mediante dos puntos (`..`).

- `cd [<ruta>]`: permite cambiar de directorio. Por ejemplo `cd /home/pepito/` permite movernos al directorio `home` del usuario `pepito`. El comando `cd ..` me lleva al directorio padre del directorio actual. El comando `cd` me lleva al directorio `home` del usuario.

- `rm [<ruta>]<nombre_fichero>`: permite borrar un fichero cuyo nombre (y/o ruta completa) se indica. Es necesario emplear este comando con cuidado, dado que un fichero que se borra con este comando no puede recuperarse posteriormente.

- `mkdir [<ruta>]<nombre_directorio>`: permite crear un directorio cuyo nombre (y/o ruta completa) se indica. Ejemplo: `rmdir /home/pepito/temp`.

- `rmdir [<ruta>]<nombre_directorio>`: permite borrar un directorio, si está vacío, es decir, si no contiene ficheros.

- `cp [<ruta>]<nombre_fichero> [<ruta>][<nombre_fichero>]`: copia un fichero en una ruta concreta.
  Por ejemplo, `cp /home/pepito/programas/hola_mundo.cpp /home/pepito/temp` copia el fichero `hola_mundo.cpp` del directorio `/home/pepito/programas` al directorio `/home/pepito/temp`.

- `mv [<ruta>]<nombre_fichero> [<ruta>][<nombre_fichero>]`: mueve un fichero a una ruta concreta.
  Por ejemplo, `mv /home/pepito/programas/hola_mundo.cpp /home/pepito/temp` mueve el fichero `hola_mundo.cpp` del directorio `/home/pepito/programas` al directorio `/home/pepito/temp`.

- `cat <nombre_fichero_texto>`: muestra por pantalla el contenido de un fichero de texto.

1.2 Herramientas de programación en C++ bajo Linux

Una vez analizados las características básicas y los constituyentes principales del sistema operativo Linux, que nos servirán para el desarrollo de las prácticas en el laboratorio, a continuación se expone una serie de nociones básicas sobre las herramientas necesarias para el desarrollo de programas en C++: el editor, el compilador y el depurador. El objetivo de esta sección es introducir los elementos mínimos que permitan escribir, compilar y depurar los programas que se desarrollarán en los siguientes capítulos de la asignatura. Durante el curso, se irán añadiendo progresivamente los elementos necesarios para aumentar el rendimiento y las posibilidades de estas herramientas.

1.2.1 El editor GVim

GVim es un editor de texto que permite múltiples opciones de configuración para la edición eficiente de texto. Es una versión mejorada del clásico editor *vi*, a la que se han añadido nuevas utilidades y la posibilidad de manejar las funciones del editor mediante

el empleo de menús y barras de botones (véase figura 1.4). GVim permite la edición avanzada y eficiente de programas en múltiples lenguajes de programación, entre ellos C++. Algunas de las características que lo dotan de gran utilidad son las siguientes:

- Coloreado de la sintaxis (*syntax highlight*), que permite resaltar en diferentes colores los distintos elementos del programa: comentarios, palabras reservadas, cadenas de caracteres, etc.
- Auto-indentación del texto.
- Múltiples *buffers*: permite mantener abiertos varios ficheros de texto para la edición y pasar de uno a otro con facilidad.
- Múltiples ventanas: permite dividir la ventana de edición en múltiples ventanas verticales y/o horizontales.
- Utilidades de edición: búsqueda de palabras o expresiones, sustitución, múltiples opciones para situarse en el texto, definición de macros, definición de abreviaturas, etc.
- Ayuda sobre los comandos y las opciones del editor.

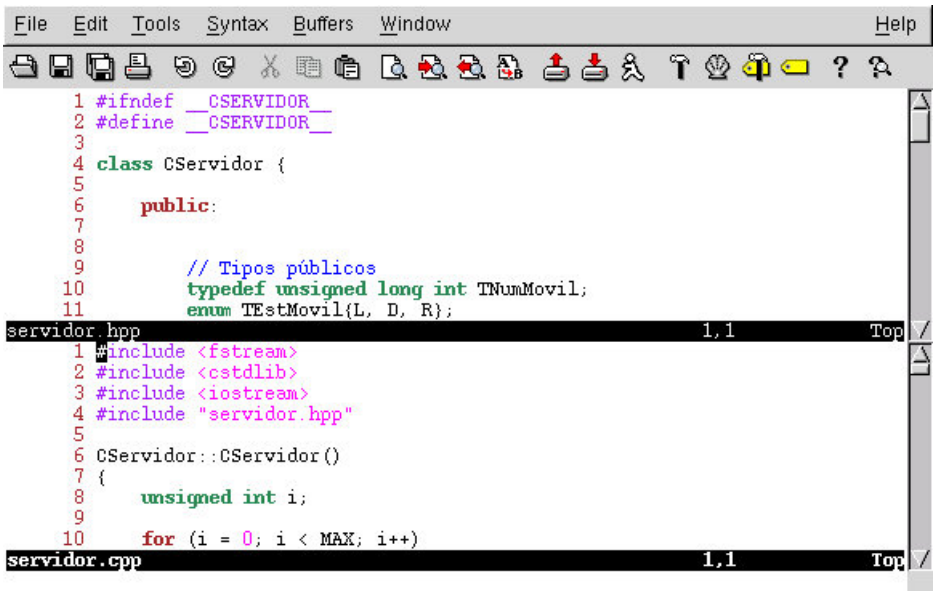

Figura 1.4: El editor GVim

Para ejecutar el editor, basta con teclear en un *shell* el comando `gvim`, o bien `gvim nombre.cpp` para editar el texto `nombre.cpp` (la extensión `.cpp` es una de las posibles

extensiones de los ficheros fuente en C++). También es posible lanzar el editor a través de la correspondiente selección en el menú K, por ejemplo: `Menú K->Editores->Gvim`.

Lo primero que destaca en el manejo del editor GVim es la existencia de dos modos de trabajo: modo *inserción* y modo *comando*. En el momento de lanzar el editor, éste se encuentra en modo comando, de manera que no es posible introducir o modificar texto, aunque sí desplazarse por él con las correspondientes teclas (flechas, página arriba o abajo). Para entrar en modo inserción basta con introducir el comando `i`, o bien pulsar la tecla `<Insert>` en el teclado extendido. La escritura, eliminación y movimiento por el texto puede hacerse mediante las teclas habituales empleadas en otros editores de Linux o Windows.

Para salir del modo inserción es necesario pulsar la tecla `<ESCAPE>`, entrando así en modo comando. La mayoría de los comandos comienzan con el carácter `:`, es decir, 'dos puntos'. Por ejemplo, para guardar un fichero editado bajo el nombre `holamundo.cpp` se emplea el comando `:w holamundo.cpp`. Para salir del editor, una vez que se ha grabado el fichero en disco, basta con teclear, en modo comando, `:q`.

Todas las utilidades del editor (grabación, apertura de un fichero de disco, búsqueda y sustitución, etc.) pueden realizarse sin necesidad de conocer los comandos del editor, usando en su lugar los botones de la barra de herramientas o las opciones del menú (véase figura 1.4). No obstante, dado lo extendido que está el editor GVim (sobre todo en sus versiones no gráficas *vi* o *vim*, que requiere el uso de comandos, en ausencia de botones o menús) conviene aprender algunos de los comandos más habituales:

- `i`: inserta en la posición actual del cursor (antes del carácter sobre el que está situado éste).
- `a`: inserta después del cursor (después del carácter donde está situado éste).
- `I`: inserta al principio de la línea.
- `A`: inserta al final de la línea.
- `o`: introduce una línea a continuación de la actual y se pone el editor en modo inserción.
- `O`: introduce una línea antes de la línea actual y se pone en modo inserción.
- `gg`: se sitúa al principio del texto (sin entrar en modo inserción)
- `G`: se sitúa al final del texto (sin entrar en modo inserción).

- `$`: sitúa el cursor al final de la línea actual (sin entrar en modo inserción).
- `:q`: sale del editor.
- `:w`: graba el buffer (fichero) actual.
- `:wq`: graba el buffer actual y sale del editor.
- `:q!`: sale del editor sin grabar (fuerza la salida aunque haya habido cambios que no se han grabado aún).
- `:s/expresión`: busca la siguiente aparición de `expresión` en el texto. También resalta en amarillo todas las apariciones de `expresión`.
- `:s/antigua/nueva`: sustituye la siguiente aparición de `antigua` por `nueva`.
- `n`: sitúa el cursor en la siguiente aparición de la expresión buscada en la última búsqueda.
- `N`: sitúa el cursor en la anterior aparición de la expresión buscada en la última búsqueda.
- `u`: deshace la última acción de edición realizada.
- `dd`: borra la línea completa sobre la que está actualmente el cursor.
- `d$`: borra desde la posición actual del cursor hasta el final de la línea.
- `:set nu`: imprime el número de línea al principio de cada línea, de manera automática.
- `:help`: muestra ayuda sobre el editor.
- `:help <comando>`: muestra ayuda sobre el comando que se le pasa como parámetro.

1.2.2 El compilador GCC

El compilador usado en las prácticas es el GCC 3.x. Las siglas GCC significan *GNU Compiler Collection*, es decir, 'colección de compiladores de GNU', en referencia a los diferentes compiladores que se engloban bajo el nombre GCC. El compilador de C++ es uno de ellos.

El proceso de compilación y enlazado de programas está esquematizado en la figura 1.5. La compilación de un programa (habitualmente llamado *código fuente*) implica, a

grandes rasgos, la traducción de un programa escrito en un lenguaje de programación de alto nivel (como es el caso de C++) en un fichero binario ejecutable por el ordenador. Durante el proceso de compilación, el compilador puede detectar errores sintácticos o semánticos en el programa que impidan la correcta generación de un ejecutable. Los errores y avisos (*warnings*) del compilador se muestran por pantalla (habitualmente) y consisten en mensajes destinados a que el programador corrija el fichero fuente. Una vez que el programa está libre de errores, el compilador puede generar un fichero ejecutable. La existencia de *warnings* por parte del compilador, sin presencia de errores, no impide, sin embargo, la generación del ejecutable.

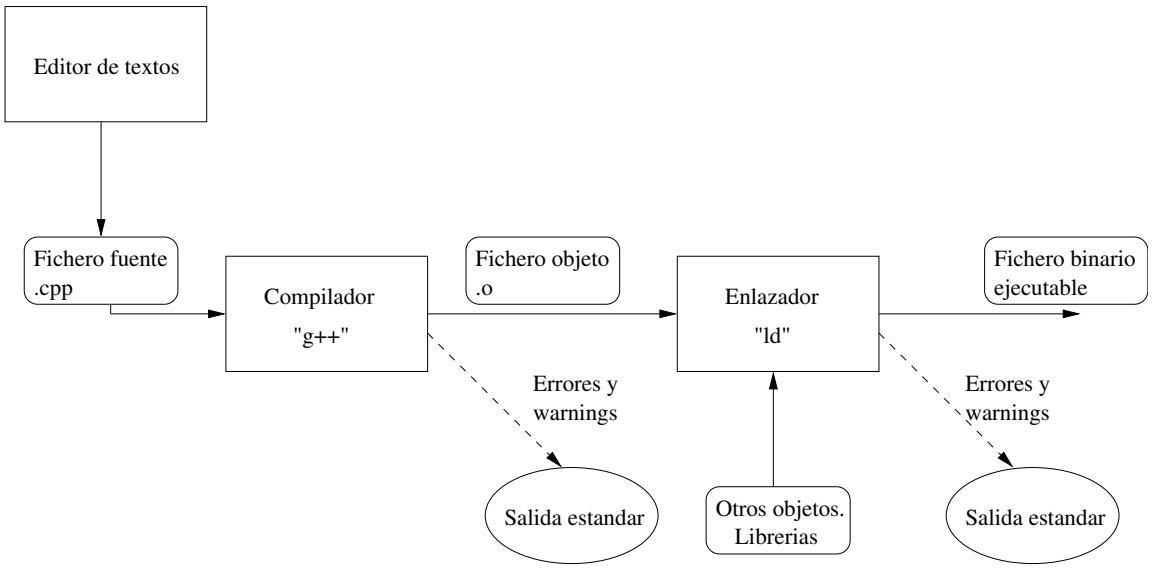

Figura 1.5: El proceso de compilación y enlazado de programas

No obstante, la generación de un fichero ejecutable a partir de un fichero fuente implica otro paso intermedio, como es esquematiza en la figura 1.5. Esta fase intermedia se conoce como *enlazado* o *linkado*. En realidad, la compilación de un fichero fuente correcto da como resultado un fichero intermedio, denominado *código objeto*, al que es necesario habitualmente enlazar otros objetos para obtener el fichero ejecutable. Normalmente, estos otros ficheros objeto son librerías estándar de programas que facilitan al programador la tarea de la programación: rutinas estándares de entrada/salida, librerías matemáticas, etc. El proceso de enlazado puede generar sus propios mensajes de error y avisos por parte del *enlazador* o *linker*.

Para mostrar el proceso de compilación con GCC de un programa en C++, vamos a valernos de un ejemplo simple. En el editor GVim, vamos a escribir y guardar nuestro primer programa en C++. Para ello, ejecutamos el editor mediante el comando `gvim`

holamundo.cpp (como puede comprobarse, la extensión del fichero fuente en C++ es .cpp, aunque el compilador GCC permite otras). A continuación, entramos en modo inserción (pulsamos la tecla i) y escribimos el siguiente código:

```
#include <iostream>
#include <cstdlib>

// Este es mi primer programa en C++
int main(){
    using namespace std;

    cout << "Hola mundo" << endl;

    return 0;
}
```

Una vez teclado el programa, salimos del modo inserción en GVim (pulsamos <ESCAPE>) y grabamos (:w). En este momento, se habrá creado en el directorio de trabajo actual el fichero holamundo.cpp.

El compilador GCC será usado en modo comando sobre una *shell*. El comando básico que permite compilar un programa es el siguiente:

```
g++ -c -o <fichero_objeto> <fichero_fuente>
```

El comando g++ llama al compilador. La opción -c significa 'compilar'. La opción -o permite especificar el nombre del fichero objeto de la compilación. Por último, se introduce el nombre del fichero fuente con el programa en C++. De esta manera, para compilar el programa anterior, es necesario escribir el siguiente comando:

```
g++ -c -o holamundo.o holamundo.cpp
```

Esto generará en el directorio de trabajo actual el fichero holamundo.o. Si se han cometido errores sintácticos o semánticos durante la edición del programa, el compilador mostrará por pantalla un listado de los errores, indicando el número de línea donde éstos se han producido. Para crear un fichero ejecutable, mediante una llamada al enlazador, es necesario escribir:

```
g++ -o holamundo holamundo.o
```

La llamada al enlazador es simular a la compilación, ejecutando g++, excepto en la ausencia de la opción -c, lo cual indica que se trata de un fichero objeto que hay que enlazar para generar un fichero ejecutable: holamundo. Es preciso hacer constar aquí que en Linux la extensión de los ficheros ejecutables (binarios) puede ser cualquiera (incluida

la extensión vacía), a diferencia de otros sistemas operativos. El comando anterior habrá creado en el directorio de trabajo un fichero ejecutable, que podemos invocar escribiendo su nombre precedido de la ruta: `./holamundo`. De esta manera, el 'programa' `holamundo`, que está en el directorio actual (`./`) se ejecuta y escribe por pantalla la frase `Hola mundo`.

El proceso de compilación y linkado puede también hacerse con una sola llamada al compilador, de la siguiente manera:

```
g++ -o holamundo holamundo.cpp
```

El compilador GCC permite múltiples opciones, que se especifican en la llamada a `g++`. En las páginas del manual para GCC (`man gcc`) se encuentra la ayuda completa sobre el uso del compilador, donde se especifican cada una de las opciones disponibles. Algunas de ellas, las más utilizadas son las siguientes:

- `-Wall`: muestra, en caso necesario, todos los avisos (*warnings*) posibles.
- `-ansi`: soporte para todos los programas en C ANSI.
- `-Werror`: convierte los avisos en errores para impedir que el programa genere un binario en caso de que existan avisos.
- `-pedantic`: muestra todos los avisos demandados por el ANSI C de manera estricta.

Por ejemplo, el programa anterior se podía haber compilado (y enlazado) de la siguiente manera:

```
g++ -o holamundo -Wall -Werror -ansi -pedantic holamundo.cpp
```

1.2.3 El depurador de código GDB/DDD

Como se ha comentado en el apartado 1.2.2, el compilador muestra por pantalla los errores y *warnings* que se generan durante el proceso de compilación de un programa, como consecuencia de los errores introducidos en el código. Sin embargo, un programa que compile sin errores ni *warnings* puede no ser correcto, en el sentido de no hacer, cuando se ejecuta, aquello para lo que estaba diseñado. Esto quiere decir que el programa, en su ejecución, no se ajusta a las especificaciones de su diseño.

Un depurador (*debugger*) es una herramienta que permite seguir, paso a paso, la ejecución de un programa para localizar posibles incorrecciones en su implementación. Para ello, un depurador debe permitir ejecutar las instrucciones paso a paso, detenerse en una determinada instrucción, visualizar valores de variables y expresiones, etc.

GDB (*GNU DeBugger*) es uno de los depuradores más extendidos para el desarrollo de programas bajo UNIX y Linux. Funciona en modo comando, esto es, es preciso aprender

un conjunto de órdenes para llevar a cabo las tareas de depuración de un ejecutable. Sin embargo, existen algunas aplicaciones que facilitan la comunicación con el depurador GDB mediante el empleo de una interfaz gráfica. Este es el caso de DDD (*Data Display Debugger*), que constituye una interfaz (*front-end*) gráfica para GDB y otros depuradores que funcionan con líneas de comandos. Las principales tareas que pueden realizarse con DDD son las siguientes:

- Ejecutar un programa, especificando todo lo que pueda afectar a su comportamiento.
- Hacer que el programa se detenga bajo ciertas condiciones.
- Examinar qué ha ocurrido cuando un programa se ha detenido.
- Cambiar 'cosas' en el programa, de manera que se pueda experimentar, corrigiendo los efectos de algún error del programa para poder pasar al siguiente error.

Lo primero que se necesita para poder depurar un programa es que éste haya sido compilado indicando la opción de generación de información para la depuración. Con el compilador GCC, esto se consigue mediante la opción `-g`. Por ejemplo, podemos compilar el programa del apartado 1.2.2, mediante el comando:

```
g++ -g -o holamundo holamundo.cpp
```

A continuación, podemos lanzar el depurador DDD para depurar el programa `holamundo`. Para ello, basta con escribir en el *shell* el comando `ddd holamundo`. En la figura 1.6 puede verse el aspecto del depurador DDD que se ha lanzado para la depuración del programa `holamundo`. Como puede comprobarse en la figura, la presencia de botones, menús y barras de herramientas hace más confortable la depuración del programa. En el ejemplo de la figura se ha introducido un *punto de ruptura* (que se muestra como una señal de stop) en una de las líneas del programa, lo que hará que el programa se detenga en esa línea cuando se lance su ejecución.

1.3 Algunos enlaces importantes

Sobre el sistema operativo Linux/GNU

- `http://www.linuxiso.org/`
  Podemos encontrar enlaces para bajar las principales distribuciones de Linux.
- `http://www.redhat.com/`
  Distribución de Linux RedHat.

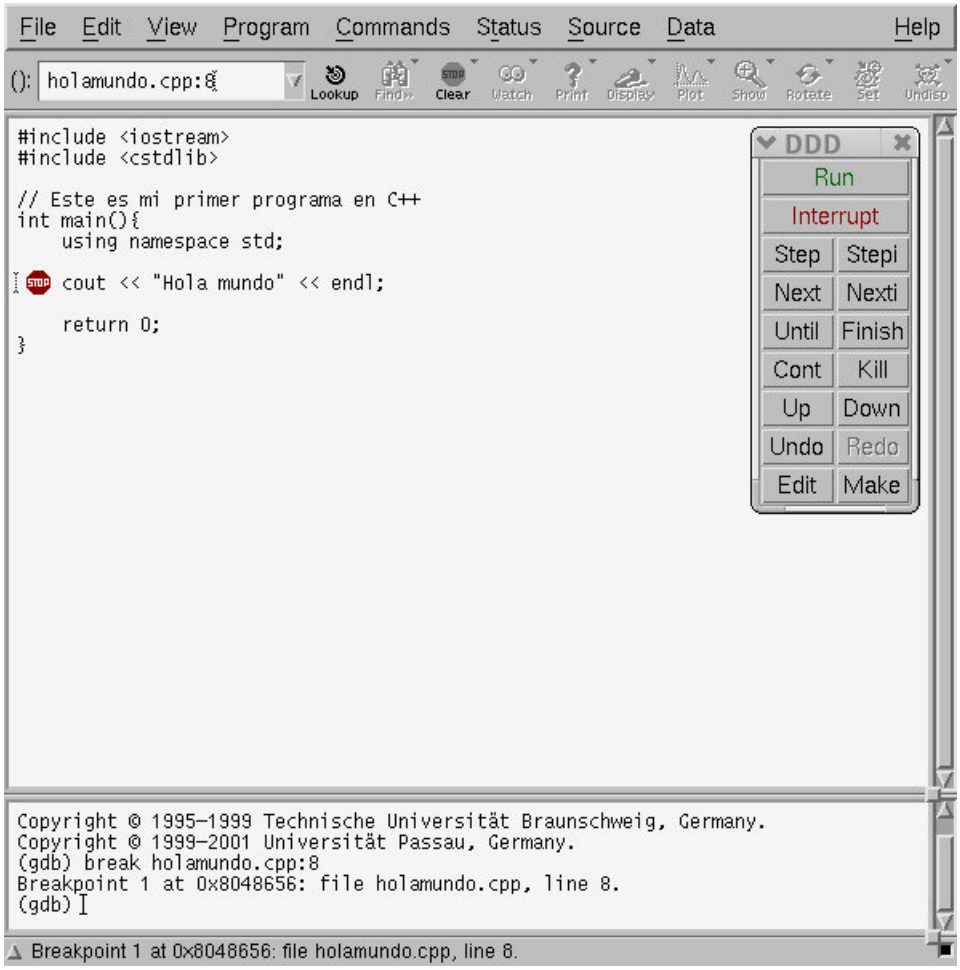

Figura 1.6: El depurador de código DDD

- `http://www.knopper.net/knoppix/index-old-en.html`
  Distribución de Linux Knoppix.

- `http://www.gnu.org/`
  Página del proyecto GNU para desarrollo de software gratuito de código abierto.

Programación en C++ bajo Linux:

- `http://gcc.gnu.org/`
  Página del compilador GCC de GNU.

- `http://www.vim.org/`
  Sitio web del editor Vim (GVim).

- `http://sources.redhat.com/gdb/`
  Sitio web sobre el depurador de programas GDB.

- `http://www.gnu.org/software/ddd/`
  Páginas sobre la interfaz gráfica para la depuración DDD.

- `http://www.gnu.org/manual/glibc-2.2.3/html_chapter/libc_toc.html`
  Ayuda sobre la implementación de GNU (GCC) de la librería estándar de C.

- `http://www.cplusplus.com/`
  Páginas sobre ayudas y recursos para la programación en C++.

1.4 Notas sobre la distribución Linux/Knoppix

- *Configurar la BIOS del PC.*

  Para ejecutar el software desde el CDROM hay que configurar la BIOS. Esto se hace, usualmente, pulsando la tecla de `<Supr>` al arrancar del ordenador, (F2 en portátiles). Si no fuera posible arrancar desde el CD ROM, hay que crear un disquete de arranque con la imagen `boot.image` del directorio KNOPPIX del CDROM (también es posible solicitar el envio de esa imagen desde la web de Knoppix). Para hacer esto en Linux, se debe usar el comando `dd if=/cdrom/KNOPPIX/boot.img of=/dev/fd0`. Desde Windows, se puede usar el programa `rawrite`, incluido en ese directorio.

- *Arrancando.*

  Una vez arrancado el PC desde el CDROM, aparecerá en pantalla la presentación de Knoppix y la palabra `boot:`. Pulsando simplemente la tecla `<ENTER>`, basta esperar un par de minutos hasta tener el gestor de ventanas KDE3 funcionando. Durante este tiempo, se pueden mirar los mensajes de autodetección del hardware.

  Para que arranque el sistema operativo usando el lenguage español, se puede escribir en el `boot: knoppix lang=es` (cuidado, el teclado inicialmente está en aleman, por lo habrá que localizar con cuidado los caracteres, como el carácter '='). No obstante, una vez iniciada la sesión KDE, puede cambiarse el lenguaje de trabajo pulsando el botón derecho del ratón sobre la bandera que aparece en la parte derecha de la barra de herramientas de KDE, y pulsando en 'Configurar'. De esta manera se puede seleccionar el lenguaje deseado.

  Si después de los mensajes de arranque la pantalla se queda negra, el problema puede radicar en la frecuencia de refresco soportada por le monitor. En este caso, se debe rearrancar apretando el botón RESET del PC o pulsando simultáneamente

las teclas `<CNTRL>+<ALT>+<SUPR>`. Cuando el sistema rearranque y nos presente el `boot`, se puede introducir lo siguiente: `boot: knoppix xvrefresh=60`. Si el monitor soporta más de 60Hz de frecuencia de refresco (debe verse el manual del monitor), introducir `boot: knoppix xserver=fbdev`.

Si dispone de ratón con ruedecita, para activarlo puede arrancarse con `boot: knoppix wheelmouse`.

El resto de las opciones de arranque puede visualizarse pulsando `<F2>` cuando aparece el `boot:`. Esto es un pequeño resumen de estas opciones (las opciones completas (*cheatcodes*) pueden encontrarse en *http://www.linuxtag.org/knoppix/knoppix-cheatcodes.txt*:

knoppix lang=cn—de—da—es—fr—it—nl	specify language/keyboard
knoppix lang=pl—ru—sk—tr—tw—us	specify language/keyboard
knoppix screen=1280x1024	Use specified Screen resolution for X
knoppix xvrefresh=60 (or vsync=60)	Use 60 Hz vertical refresh rate for X
knoppix xhrefresh=80 (or hsync=80)	Use 80 kHz horizontal refresh rate for X
knoppix xmodule=ati—fbdev—i810—mga	Use specified XFree4-Module (1)
knoppix xmodule=nv—radeon—savage—s3	Use specified XFree4-Module (2)
knoppix xmodule=radeon—svga—i810	Use specified XFree4-Module (3)
knoppix 2	Runlevel 2, Textmode only
knoppix usb2	Try to initialize USB 2.x controller(s)
knoppix pci=irqmask=0x0e98	Try this, if PS/2 mouse doesn't work *)
knoppix mem=128M	Specify Memory size in MByte
knoppix noeject	Do NOT eject CD after halt
knoppix noprompt	Do NOT prompt to remove the CD
knoppix vga=normal	No-framebuffer mode, but X
knoppix nowheelmouse	Force plain PS/2 protocol for PS/2-mouse
fb1280x1024	Use fixed framebuffer graphics (1)
fb1024x768	Use fixed framebuffer graphics (2)
fb800x600	Use fixed framebuffer graphics (3)
knoppix keyboard=us xkeyboard=us	Use different keyboard (text/X)
expert	Interactive setup for experts

- *Guardando su configuración para sesiones futuras.*

  Puede guardarse la configuración del sistema para próximas sesiones, de manera que no haya que reconfigurar cada vez que arranca Knoppix. Para ello, hay que seleccionar `Menú K->KNOPPIX->Configuración->Guardar configuración de KNOPPIX`. Se puede elegir dónde guardar la configuración: en un disquete o en una de las particiones (por ejemplo, Windows) de que disponga el ordenador.

Para recuperar la configuración al rearrancar debe de escribir en el `boot: knoppix myconfig=scan`. Con esta opción de arranque, se escanearán las particiones (incluído el disquete) para intentar localizar un fichero de configuración. También puede indicarse directamente la ruta de arranque. Por ejemplo, con `boot: myconfig=/dev/fd0` se carga la configuración desde el disquete.

- *Guardando su trabajo para futuras sesiones.*

  Cuando arranca el sistema operativo Knoppix, se crean en memoria RAM particiones de disco virtuales (no permanentes). El usuario `knoppix` (que es el único usuario que posee el sistema, de entrada, además de `root`) escribirá normalmente en su directorio `/home/knoppix/`, que será el directorio de trabajo habitual. Sin embargo, es necesario guardar todos los datos en alguna unidad permanente para que no se pierdan cuando se apague el sistema. Es posible guardar los datos en un disquete o en cualquiera de las particiones de disco que posee el sistema, si estas son de tipo FAT16, FAT32, EXT2 o EXT3. Sobre las particiones NTFS **no está recomendado escribir**. Por defecto, las particiones de disco se montan sólo de lectura. Para montarlas para lectura/escritura basta con pulsar el botón derecho del ratón sobre el icono de la partición en el escritorio de KDE y deshabilitar la opción de 'sólo lectura' de la ficha de 'Dispositivo'. La copia de datos a las particiones de disco o disquetes pueden realizarse de la manera habitual, por ejemplo utilizando el navegador *Konqueror*.

  Una buena alternativa a la escritura en las particiones de disco es el uso de dispositivos tipo *pen drive*, que se conectan a los puertos USB del ordenador y que son detectados y gestionados de manera automática y eficaz por el sistema Knoppix.

  Durante el arranque, Knoppix detecta las particiones presentes en su sistema. En el escritorio KDE aparecerá un icono por cada una de las particiones detectadas por Knoppix. Si el ordenador posee puertos USB y Knoppix los ha detectado y configurado pertinentemente, cuando inserte un dispositivo USB de almacenamiento (por ejemplo, un *pen drive* o *key drive*) aparecerá también en pantalla el icono correspondiente. Para activar una partición, un disco flexible o un dispositivo de almacenamiento USB, es necesario pulsar el botón derecho del ratón sobre el icono correspondiente en el escritorio y 'Montar' la partición antes de proceder a la lectura o escritura de los ficheros de esa partición.

  Es posible también crear un sistema de ficheros permanente (directorio `/home/knoppix`) para todas las siguientes sesiones de Knoppix, de manera que no sea necesario

copiar datos a otras particiones/dispositivos antes de cerrar la sesión de Knoppix. Una buena opción para hacer esto es, una vez más, un dispositivo externo tipo *pen drive*. Para crear este sistema de ficheros permanente es necesario ir a `Menú K -> KNOPPIX -> Configuración -> Create a persistent KNOPPIX home directory` y seguir los pasos que se indican. Es posible crear ese directorio en cualquier partición de disco duro o dispositivo de almacenamiento. [1].Durante el arranque, puede indicarse a Knoppix dónde está el directorio `home` persistente mediante la opción de arranque (*boot*) `home=/dev/sda1`, si se ha activado el directorio persistente en un *pen drive* USB. También es posible indicar a Knoppix que realice una búsqueda automática de dispositivos de almacenamiento, con la opción `home=scan`.

1.5 El compilador GCC sobre Windows

MinGW es una colección de bibliotecas y programas de aplicaciones de GNU que han sido traducidas para crear programas en Windows. Entre estas aplicaciones se encuentra el compilador GCC.

Esta colección de programas está disponible gratuitamente en la página web

http://www.mingw.org

Para conseguirlo basta con descargar el fichero que está disponible en

http://prdownloads.sf.net/mingw/MinGW-3.1.0-1.exe?download

Para instalarlo basta con ejecutarlo e indicarle en qué directorio queremos instarlarlo. Es recomendable que el camino del directorio de instalación no incluya espacios en blanco ni caracteres acentuados. Una vez instalado es necesario incluir el directorio `DirInstal\bin` en la variable `PATH` del sistema para poder ejecutar el compilador desde cualquier directorio. `DirInstal` se refiere al directorio que se ha escogido para la instalación.

El manejo es exactamente igual que la versión de Linux.

[1] Ojo, si se elige la opción de usar la partición completa (*ENTIRE partition*) para guardar el directorio *home* persistente, se perderán los datos de esa partición, dado que Knoppix formatea la partición entera bajo el sistema de ficheros de linux (*ext2*). Por este motivo, se recomienda elegir la opción de crear una **imagen** del sistema de ficheros para almacenar el directorio permanente.

Bibliografía

[Gri02] A. Griffith. *GCC The Complete Reference*. McGraw-Hill, 2002.

[Oua01] S. Oualline. *Vi iMproved (VIM)*. SAMS, 2001.

[Pet02] R. Petersen. *Linux: manual de referencia*. McGraw-Hill, 2 edition, 2002.

[SKS03] S.M. Sarwar, R. Koretsky, and S.A. Sarwar. *El Libro de LINUX*. Addison Wesley, 2003.

[SPSS02] R. Stallman, R. Pesch, S. Shebs, and R.M. Stallman. *Debugging with GDB: The GNU Source-Level Debugger*. Free Software Fundation, 2002.