1. El compilador: gcc

Compilador gratuito de C/C++ del proyecto GNU.
Sintaxis:
gcc ***[opciones] f1.c f2.c ...*** **-L*lib1 ...***
gcc ***[opciones] f1.c f2.c f3.o f4.o ...*** **-L*lib1 ...***

- Al compilador se le puede llamar también **cc**.
- Salvo indicación en contrario, se genera el ejecutable **a.out**
- Si la extensión del fichero es **.cc**,**.C**,**.cxx** o **.cpp**, compilará en lenguaje C++.

Opciones más importantes

-g	genera información para el depurador
-c	genera sólo los objetos **.o** Si se pasan ficheros **.o** como argumentos, no se hace nada con ellos
-o ***fichero***	el ejecutable se llamará ***fichero***, en lugar de **a.out**
-E	aplica sólo la fase del preprocesador
-S	genera sólo los fuentes en ensamblador **.s**
-O	optimiza el código máquina generado
-O3	optimización brutal
-l*biblio*	enlaza la biblioteca **lib*biblio*** (ej. **-lm**, biblioteca matemática)
-I*ruta*	añade ***ruta*** a las rutas de búsqueda para resolver los **#include <*fichero.h*>**
-L*ruta*	añade ***ruta*** a las rutas de búsqueda de bibliotecas
-D*símbolo*	define la macro ***símbolo*** (equivale a escribir **#define** ***símbolo*** en todos los fuentes)
-D*sim=valor*	equivale a escribir en todos los fuentes **#define** ***sim valor***
-fsyntax-only	no compila, sólo analiza el prg.
-v	muestra los pasos de la compilación

2. El depurador: gdb

El **gdb** es el depurador gratuito elaborado por GNU. Existen otros depuradores en UNIX (dbx, xdb), de pago, cuyo comportamiento es similar al GDB, aunque los nombres de las órdenes varían un poco.
Puede leerse una guía completa del ***gdb*** invocando desde el *shell* la orden **info gdb**

2.1 Invocación al depurador gdb

Para poder depurar un programa, el ejecutable se tiene que haber compilado usando la opción **-g** del compilador. De lo contrario, se verá sólo código máquina.

Depurar un programa:
gdb ***archivo_ejecutable***

Depuración *post-mortem*:
gdb ***ejecutable fichero_core***
Permite depurar un programa que ha abortado. Cuando un programa aborta en UNIX, suele volcar su estado en un archivo llamado **core**.

Capturar un proceso en ejecución:
gdb ***ejecutable NNNN***
Interrumpe el proceso cuyo pid es ***NNNN*** y lo pasa a depurar. El proceso tiene que pertenecer al usuario que está depurando.

Algunas opciones del **gdb**:

-h	(*help*) Ayuda rápida sobre el depurador.
-d ***ruta***	Incluye a ***ruta*** como directorio donde buscar fuentes. Útil si algunos fuentes están en otro directorio.
-q	(*quiet*) No aparecen mensajes de bienvenida cuando arranca el depurador.

2.2 Dar órdenes al depurador

Las órdenes se escriben en la parte inferior de la pantalla, después de donde pone **(gdb)**
Ejemplo: **break main** pone un punto de ruptura en la rutina ***main***.

- Los nombres de las órdenes se pueden abreviar si no hay ambigüedad, o con las órdenes más frecuentes. p.ej. **b** en vez de **break**. En este tríptico se usan sobre todo los nombres abreviados.
- Si se escribe sólo un salto de línea, se repite la última orden tecleada.
- Si se escriben las primeras letras de una función o variable y luego se pulsa **TAB**, el ***gdb*** intenta rellenar el resto del nombre.
- Lo mismo ocurre si escribimos las primeras letras de una orden del ***gdb***.

2.3 Cómo indicar direcciones

Cuando en la explicación de una orden aparezca el argumento ***sitio***, se refiere a una posición en el fuente, que puede expresarse de estas formas:

línea	Número de línea dentro del fichero fuente actual
fich.c:línea	Una línea dentro del fichero ***fich.c***
proc	La primera línea de la subrutina ***proc***
fich.c:proc	La subrutina ***proc*** del fichero ***fich.c***
+*num*	***num*** líneas por debajo de la actual
-*num*	***num*** líneas por encima de la actual

Ejemplos
(gdb) b main:3
Pone un punto de ruptura en la línea 3 de la función **main**
(gdb) l fichero.c:15
Visualiza a partir de la línea 15 del fuente **fichero.c**

2.4 Identificadores de variables

En las órdenes que manejan datos (ej. la orden **p**), éstos pueden escribirse así:

var	La variable ***var*** del procedimiento actual
proc**::*****var***	La variable ***var*** del procedimiento ***proc***
::*var*	La variable global ***var***

2.5 Ejecución del programa

r	(***run***) Comienza la ejecución del

programa, sin argumentos.

r ***args*** Comienza la ejecución del programa, con una lista de argumentos.

c (***continue***) Continúa la ejecución, si el programa está detenido

s (***step***) Ejecuta una sentencia y se detiene.

s ***num*** Ejecuta paso a paso ***num*** sentencias.

n ***[num]*** (***next***) Como **s**, pero sin entrar dentro de llamadas a procedimientos.

u (***until***) Salta a la siguiente línea (si estamos en un bucle, salimos de él)

u ***sitio*** Ejecuta hasta ***sitio***

k (***kill***) Mata el programa en ejecución

2.6 Puntos de ruptura (*breakpoints* y *watchpoints*)

Se puede forzar a que el programa se detenga en una línea cualquiera cuando trate de ejecutarla. A esto se le llama poner un *punto de ruptura (breakpoint)* en dicha línea. Además, los *watchpoints* permiten detener el programa cada vez que se lee o escribe una variable.

b ***[sitio]*** (***break***) Pone un punto de ruptura en la línea actual, o en ***sitio***.

tb ***[sitio]*** Pone un p.r. que sólo funcionará una vez

watch ***var*** Detiene el programa e imprime el valor de la variable ***var*** cada vez que se modifique

awatch ***var*** Detiene el programa cada vez que se lea o escriba en ***var***

info b Lista todos los puntos de ruptura. Cada punto de ruptura tiene un número que lo identifica.

condition ***num expr***

el p.r. ***num*** sólo funcionará cuanto ***expr*** sea cierta.

d (***delete***) Borra todos los puntos de ruptura

d ***num*** Elimina el p.r. identificado por ***num***

clear ***[sitio]*** Elimina el p.r. situado en ***sitio***

disable ***num*** Deja inactivo el p.r. ***num***

enable ***num*** Vuelve a activar el p.r. ***num***

2.7 Visualización del código fuente

l (***list***) Mira el fuente a partir de la instrucción que se está ejecutando

l ***sitio*** Mira el fuente a partir de ***sitio***

search ***texto*** Busca ***texto*** en el fuente

search Repite la última búsqueda

path ***dir*** Añade ***dir*** a las rutas donde puede haber ficheros fuentes

2.8 Ver y modificar datos del programa

i locals (***info***) Lista todas las variables locales del procedimiento actual.

p ***expr*** (***print***) Imprime el contenido de la expresión

p ***var=val*** Asigna un valor a una variable

p/x ***expr*** Imprime la expresión en hexadecimal

p/c ***expr*** La imprime como un carácter

p Imprime la última expresión escrita

La expresión ***expr*** puede ser una variable, o una expresión propia del lenguaje que se depura (empleando operadores, acceso a campos de una estructura, etc.)

2.9 Estado de la pila

En la pila aparece la secuencia de llamadas a procedimiento que se han hecho hasta llegar al punto actual de ejecución.

bt ***[N]*** (***backtrace***) Visualiza la pila hasta ***N*** niveles de profundidad

bt ***-N*** visualiza los ***N*** niveles más lejanos de la pila

2.10 Otras órdenes

quit Sale del depurador

help Ayuda general

h ***orden*** Da ayuda sobre la ***orden***

shell ***orden*** Ejecuta una orden del *shell*

file ***fichero*** Carga un nuevo ejecutable en memoria

3. Compilar y depurar en emacs

3.1 Compilar

Para compilar un fuente, escribir la orden **M-x compile** (recordar que M-x significa "escape-X"). Luego escribir el nombre del ejecutable que se va a generar, o mejor escribir la orden completa del compilador (ej. **cc -g fich.c -o fich**).

Los mensajes del compilador aparecerán en un nuevo búfer de Emacs (para dividir la ventana en dos búferes se teclea **C-x 2**).

Pulsando *intro* en un mensaje de error, el Emacs nos lleva a la línea del fuente que lo originó.

3.2 Depurar con el gdb

Se puede depurar un fuente en C escribiendo la orden **M-x gdb**. A continuación escribir el nonbre del ejecutable y las opciones para el gdb, si se desean.

El programa tiene que estar ya compilado, además con la opción **-g**.

Emacs abrirá un nuevo búfer para interactuar con el gdb. Las órdenes para el depurador se han de escribir en ese búfer. Recuérdese que para dividir la ventana Emacs en dos búferes se escribe **C-x 2**).

Si se está visualizando un archivo fuente mientras se depura, el Emacs pondrá un **=>** a la izquierda de la línea por donde va la ejecución.

4. Otro software gratuito

Otras utilidades para desarrollo de programas que pueden encontrarse en distribuciones de Linux:

xxgdb Interfaz gráfica para el GDB

xwpe Entorno integrado de compilación, al estilo del TurboPascal de Borland.