



C en GNU/Linux

Buena pareja

El lenguaje C se concibió, entre otras cosas, para realizar el sistema operativo UNIX; la mayoría de los programas del proyecto GNU se escriben en C; el núcleo Linux está escrito en C. Como se ve, el lenguaje C y el sistema operativo GNU/Linux se complementan perfectamente. Por eso, resulta muy adecuado desarrollar programas en C y aprender el lenguaje en este sistema operativo.

Editores

Para escribir el código es imprescindible un editor de textos, bien independiente o bien el editor del IDE. Ayuda mucho para escribir que el editor coloree de distinta manera cada parte del programa, lo que en inglés se llama *syntax highlighting*; muchos editores disponen de esta característica, como *emacs*, *kwrite* y *zed*.

El compilador gcc

Es casi el único que se usa en GNU/Linux, ya que es el compilador del proyecto GNU. Es habitual que la instalación del sistema deje el compilador preparado para trabajar, pero si no es así, hay que instalarlo.



Documentación

La documentación de *gcc* se encuentra en formato info, por lo que se puede leer con varios programas diferentes: *info*, *GNOME Help Browser*, *Konqueror*. Véanse dos muestras:

```

Terminal
File: gcc.info, Node: Top, Next: G++ and GCC, Up: (DIR)

Introduction
*****

This manual documents how to run, install and port the GNU compiler,
as well as its new features and incompatibilities, and how to report
bugs. It corresponds to GCC version 2.95.

* Menu:

* G++ and GCC::      You can compile C or C++ programs,
                    Command options supported by 'gcc'.
* Invoking GCC::    GNU extensions to the C language family.
* C Extensions::   GNU extensions to the C++ language.
* C++ Extensions:: gccov: a GCC test coverage program.
* Gcov::           If you have trouble installing GCC.
* Trouble::        How, why and where to report bugs.
* Bugs::          How to find suppliers of support for GCC.
* Service::       How to contribute to testing and developing GCC.
* Contributing::   Using GCC on VMS.
* VMS::

* Portability::    Goals of GCC's portability features.
* Interface::     Function-call interface of GCC output.
* Passes::        Order of passes, what they do, and what each file is for.
* RTL::          The intermediate representation that most passes work on.
* Machine Desc::  How to write machine description instruction patterns.
* Target Macros:: How to write the machine description C macros.
* Config::       Writing the 'x-MACHINE.h' file.
* Fragments::    Writing the 't-TARGET' and 'x-HOST' files.
--zz-Info: (gcc.info.gz)Top, 42 Lines --Top-- Subfile: gcc.info-1.gz
    
```

```

Terminal
File: gcc.info, Node: Invoking GCC, Next: Installation, Prev: G++ and GCC,
Up: Top

GCC Command Options
*****

When you invoke GCC, it normally does preprocessing, compilation,
assembly and linking. The "overall options" allow you to stop this
process at an intermediate stage. For example, the '-c' option says
not to run the linker. Then the output consists of object files output
by the assembler.

Other options are passed on to one stage of processing. Some options
control the preprocessor and others the compiler itself. Yet other
options control the assembler and linker; most of these are not
documented here, since you rarely need to use any of them.

Most of the command line options that you can use with GCC are useful
for C programs; when an option is only useful with another language
(usually C++), the explanation says so explicitly. If the description
for a particular option does not mention a source language, you can use
that option with all supported languages.

*Note Compiling C++ Programs: Invoking G++, for a summary of special
options for compiling C++ programs.

The 'gcc' program accepts options and file names as operands. Many
options have multi-letter names; therefore multiple single-letter options
may *not* be grouped: '-dr' is very different from '-d -r'.

You can mix options and other arguments. For the most part, the
order you use doesn't matter. Order does matter when you use several
--zz-Info: (gcc.info.gz)Invoking GCC, 68 Lines --Top-- Subfile: gcc.info-1.gz
    
```

Opciones básicas

Todos los compiladores de C admiten multitud de opciones, aunque para compilar programas sencillos no hacen falta más que unas pocas. Veamos un par de ejemplos, que serán suficientes para seguir el curso:

1. Para compilar los archivos **Fichero1.c** y **Fichero2.c** junto con la librería matemática y crear el ejecutable **Fichero**, se usa:

```
gcc -o fichero fichero1.c fichero2.c -lm
```

Como se ve, la opción **-o** sirve para indicar que el siguiente parámetro es el nombre que se desea dar al ejecutable; si no se usara, el ejecutable creado se llamaría **a.out**.

2. Para compilar los archivos **Fichero1.c** y **Fichero2.c** y crear únicamente sus archivos objeto se usa:

```
gcc -c fichero1.c fichero2.c
```

Queda de manifiesto que la opción **-c** es la que indica que sólo hay que realizar la compilación, pero no el montaje.

3. Para montar los archivos **Fichero1.o** y **Fichero2.o** y crear el ejecutable **Fichero**, se usa:

```
gcc -o fichero fichero1.o fichero2.o
```

Por tanto, es fácil separar las etapas de compilación y montaje, algo que para programas pequeños no es necesario, pero resulta imprescindible para programas grandes compuestos de muchos archivos.

La librería glibc

El lenguaje C no incluye en sí mismo capacidades que en otros lenguajes se dan por supuestas, como imprimir datos en pantalla, por ejemplo. En vez de eso, existen funciones encargadas de realizar gran cantidad de tareas. La biblioteca que reúne las funciones fundamentales de C se llama `libc`, y `glibc` es la versión GNU de `libc`; `glibc` incorpora todas las características de `libc` y algunas más, propias y exclusivas.

En general, cuando se desea realizar alguna acción en la que intervenga el sistema operativo, hay que buscar la función correspondiente consultando la documentación de `libc`, que no es la misma que la de `gcc`. Una vez que se conocen las funciones que más se necesitan, la documentación sólo se consulta de vez en cuando.

Documentación

Se encuentra en formato `info`. Véase una muestra:

```
Terminal
file: libc.info, Node: Top, Next: Introduction, Prev: (dir), Up: (dir)
Main Menu
*****
This is Edition 0.08 DRAFT, last updated 11 Jan 1999, of `The GNU C
Library Reference Manual', for Version 2.1 Beta of the GNU C Library.

* Menu:

* Introduction::          Purpose of the GNU C Library.
* Error Reporting::      How library functions report errors.
* Memory Allocation::    Allocating memory dynamically and
                        manipulating it via pointers.
* Character Handling::    Character testing and conversion functions.
* String and Array Utilities:: Utilities for copying and comparing strings
                        and arrays.
* Character Set Handling:: Support for extended character sets.
                        The country and language can affect the
                        behavior of library functions.
* Locales::              How to make the program speak the user's
                        language.
* Message Translation::  General searching and sorting functions.
                        Matching shell "globs" and regular
                        expressions.
* Searching and Sorting:: Introduction to the I/O facilities.
                        High-level, portable I/O facilities.
* Pattern Matching::     Low-level, less portable I/O.
                        Functions for manipulating files.
* I/O Overview::         A simple interprocess communication
                        mechanism.
* I/O on Streams::       A more complicated IPC mechanism, with
* Low-Level I/O::
* File System Interface::
* Pipes and FIFOs::
* Sockets::

--z-Info: (libc.info.gz)Top, 1256 lines --Top-- Subfile: libc.info-1.gz
Welcome to Info version 4.0. Type C-h for help. m for menu item.
```

```
Terminal
file: libc.info, Node: Introduction, Next: Error Reporting, Prev: Top, Up: Top
Introduction
*****
The C language provides no built-in facilities for performing such
common operations as input/output, memory management, string
manipulation, and the like. Instead, these facilities are defined in a
standard "library", which you compile and link with your programs.

The GNU C library, described in this document, defines all of the
library functions that are specified by the ISO C standard, as well as
additional features specific to POSIX and other derivatives of the Unix
operating system, and extensions specific to the GNU system.

The purpose of this manual is to tell you how to use the facilities
of the GNU library. We have mentioned which features belong to which
standards to help you identify things that are potentially non-portable
to other systems. But the emphasis in this manual is not on strict
portability.

* Menu:

* Getting Started::      What this manual is for and how to use it.
* Standards and Portability:: Standards and sources upon which the GNU
                        C library is based.
* Using the Library::    Some practical uses for the library.
* Roadmap to the Manual:: Overview of the remaining chapters in
                        this manual.

--z-Info: (libc.info.gz)Introduction, 30 lines --All-- Subfile: libc.info-2.gz
```

Programar para GNOME y KDE

La programación para entornos gráficos tiene algunas características que la hacen en principio diferente de la creación de programas para consola. Para crear un programa que tenga un interfaz GNOME o KDE, será necesario seguir ciertas reglas, utilizar algunas funciones específicas y montar el programa con librerías adicionales.

Entornos integrados

Existen en GNU/Linux varios entornos de integrados de programación. El más potente es *KDevelop*, que aunque funciona bajo KDE, permite crear programas para consola, para GNOME y para KDE. Un entorno menos potente es *Anjuta*, que funciona bajo GNOME.