Site: **FOSS Powered Wiki** at http://wiki.fosspowered.com
Source page: **The C Programming Language** at http://wiki.fosspowered.com/programming:c

The C Programming Language

Introduction

The C Programming Language is a general-purpose "high-level" programming language developed by Dennis Ritchie in order to implement the Unix Operating system for portability. C is one of the world's most popular programming language owing to its power, speed, and wide spread support community.

C is compiled language and naturally you need a compiler. There are numerous vendors which distribute C compilers. The most popular among those is the GNU C Compiler, Borland C Compiler, Microsoft Visual C++ (It's a C compiler), Digital Mars C Compiler, and many more.

Personally I recommend the GNU C Compiler (MinGW in Windows). It is an excellent compiler, free and Open Source.

In addition you need a text editor such as Gedit, Kate, or Notepad++ or the famous *nix ones such as vim and emacs.

Setting Up C Compiler (GNU C Compiler)

Depending on your operating system, it depends on how to set up you C Compiler. I will mainly focus on the popular operating systems such as Linux and Windows.

Windows

Click Here for instructions for setting up GNU C Compiler in Windows.

Ubuntu

Make sure that the **build-essential** package is installed in the system.
You can install the **build-essential** package by running the terminal and passing the command.

```
sudo apt-get install build-essential
```

Alternatively you can use the Synaptic Package Manager to search **build-essential** and install it.

OpenSUSE

Open YAST. Go to Software Management.
Change the Filter to 'Patterns' and select C/C++ Compiler and Tools.
Click Accept. And install.

Fedora

Go to terminal as root.
For running it as root, type

```
su
```

Then pass the comand.(exact as in the quotes):

```
yum groupinstall "Development Tools" "Legacy Software Development"
```

Arch Linux

The development packages for C/C++ Programming is already bundled in Arch Linux. However the package is: **base-devel**. You can use the **pacman** package manager in case any package or library is missing.

Others

See here: http://gcc.gnu.org/install/

Install an editor or IDE

For C Programming you need an editor. An simple editor would do. It should be a plain text editor and not a word processor. You can use Notepad for this purpose but MS Word or OO.org Writer is a strict no-no.
Windows users have Notepad already installed on their Systems. They can simply use Notepad for C Programming. However Notepad in itself is severely limited and barely has any features which you expect from an editor such an Syntax highlighting, indention, line numbers, etc. Thus for Windows users I highly recommend Notepad++ or Crimson Editor.
Linux users generally don't need to be taught which editors they have to use [;-)], but anyway for GNOME, **gedit** is preinstalled while for KDE. **kate** is preinstalled. **jEdit** is java based editor which is pretty decent since it's platform neutral. And there are always **vim** and **emacs** and they should probably have been first mentioned when talked about Linux and text editors.

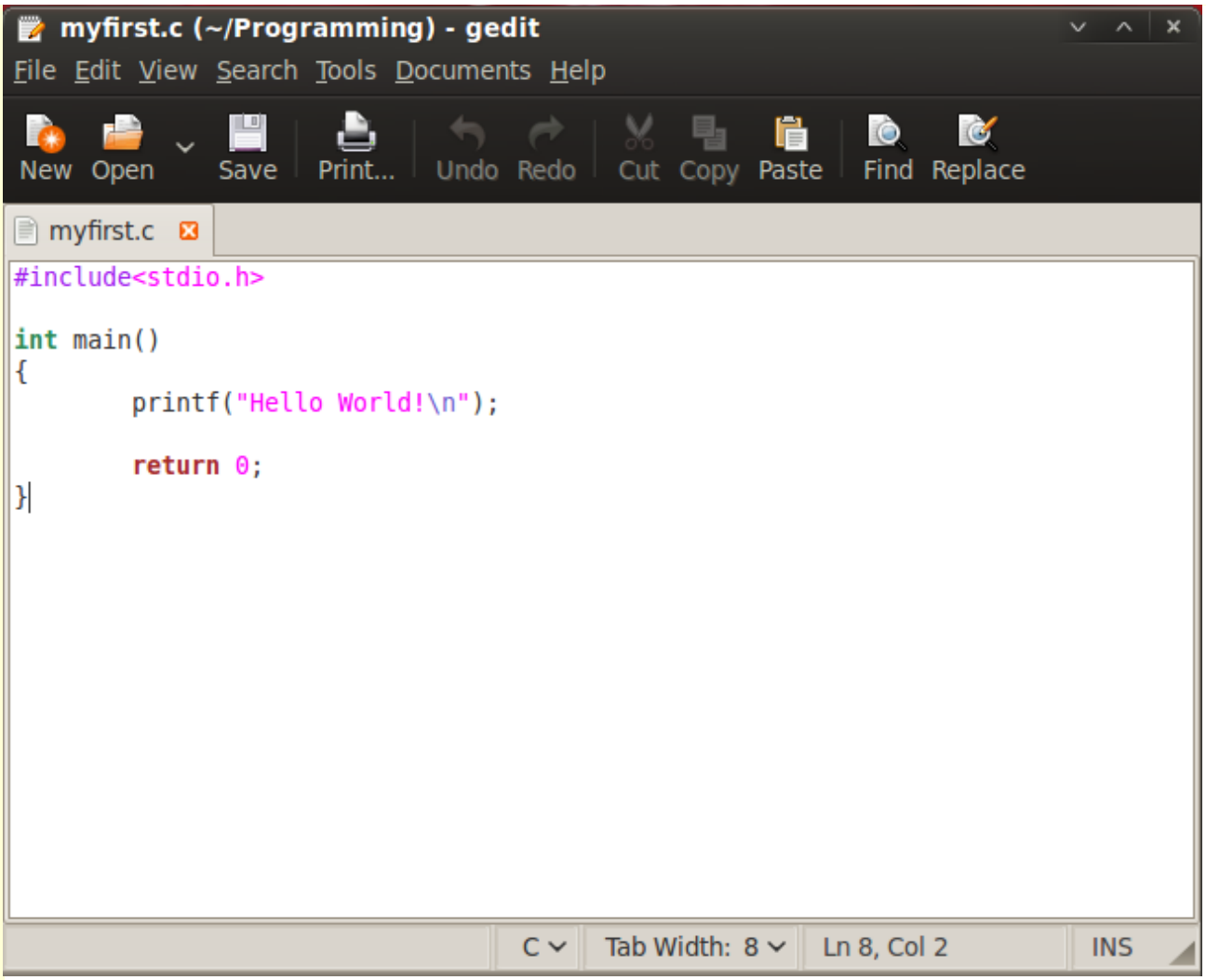
IDEs are avanced editors which offer much more than text-editing. They have extra features such as Debugging, Code Completion, and much more more. Eclipse is one such famous IDE. My recommendation is for beginner programmers is Geany. It's simple, light weight and excellent.

Compiling your first C program

Copy and paste the following into the text editor of your choice. Save it as **myfirst.c** (or any name you prefer. Make Sure the extension is .c)

```
#include <stdio.h>

int main()
{
  printf("Hello World!\n");
  return 0;
}
```



Now open the terminal/Command Prompt, go to the directory where the source file is, and compile the source file and type:

```
gcc myfirst.c
```

If all went fine and nothing is printed, it means your source file is compiled successfully so run it. The output file if you didn't specify is **a.out** :

```
./a.out
```

Hello World gets printed to the terminal.

Useful Flags for GNU C Compiler

Following are a few command line switches to enable some warnings and language features (check the man page for detailed information):

- **-std=gnu99** (C only) This flag enforces the latest C standard (1999) i.e. C99, plus GNU Extensions (this should be *explicitely* specified)
- **-ansi** This flag checks ANSI compliance
- **-pedantic** This flag issues warnings when strict ISO compatibility is NOT met.
- **-Wall** This flag enables most warnings (very useful, highly recommended)
- **-Wextra** enables more warnings (very useful)
- **-Wwrite-strings** warns when you misuse plain old C string constants (aka. deprecated cast from const char* to char*) (very useful, highly recommended)

I particularly recommend **-Wall** and **-Wwrite-strings**, to maximize warnings

and common pitfalls. Also remember paranoid programming is good for your code, so never try to ignore warnings.

Getting Help

Check out: http://gcc.gnu.org/onlinedocs/

If you are using Linux you can use Development man pages.
In Ubuntu you can install them by:

```
sudo apt-get install manpages-dev
```

Thus to get help to a function or feature. for example:

```
man 3 printf
```

```
man 3 scanf
```

To get HTML docs:

```
sudo apt-get install gcc-doc
```

And then point your browser to -> ***/usr/share/doc/gcc-doc/gcc.html***

Books and References

Check out -> http://lambdagrok.wikidot.com/learn:c

page revision: 8, last edited: