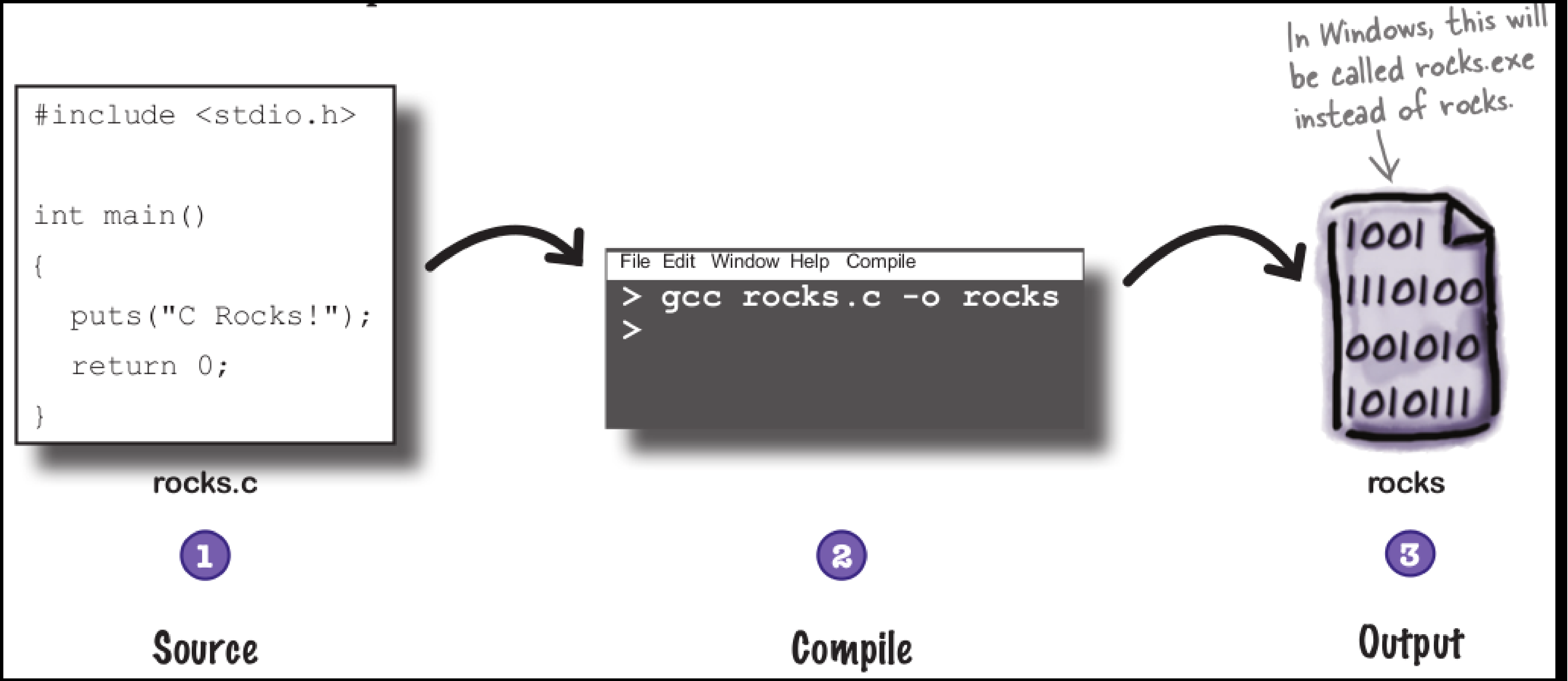
15CSE102
Computer Programming

Hello World!!

How C Works



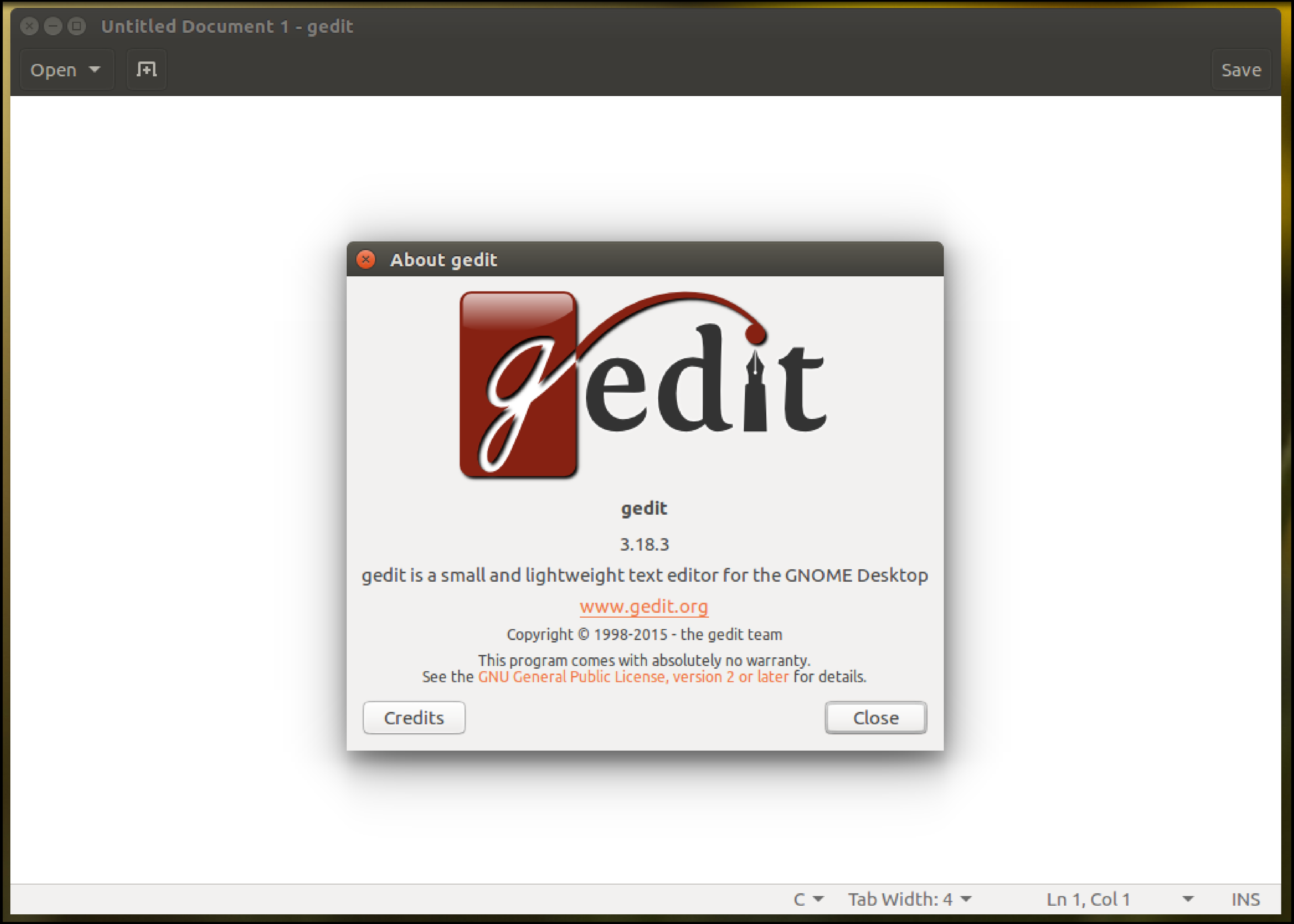
```
#include <stdio.h>

int main()
{
  puts("C Rocks!");
  return 0;
}
```

```
> gcc rocks.c -o rocks
>
```

In Windows, this will be called rocks.exe instead of rocks.

Credits: Head First C

The Editor



gedit

3.18.3

gedit is a small and lightweight text editor for the GNOME Desktop

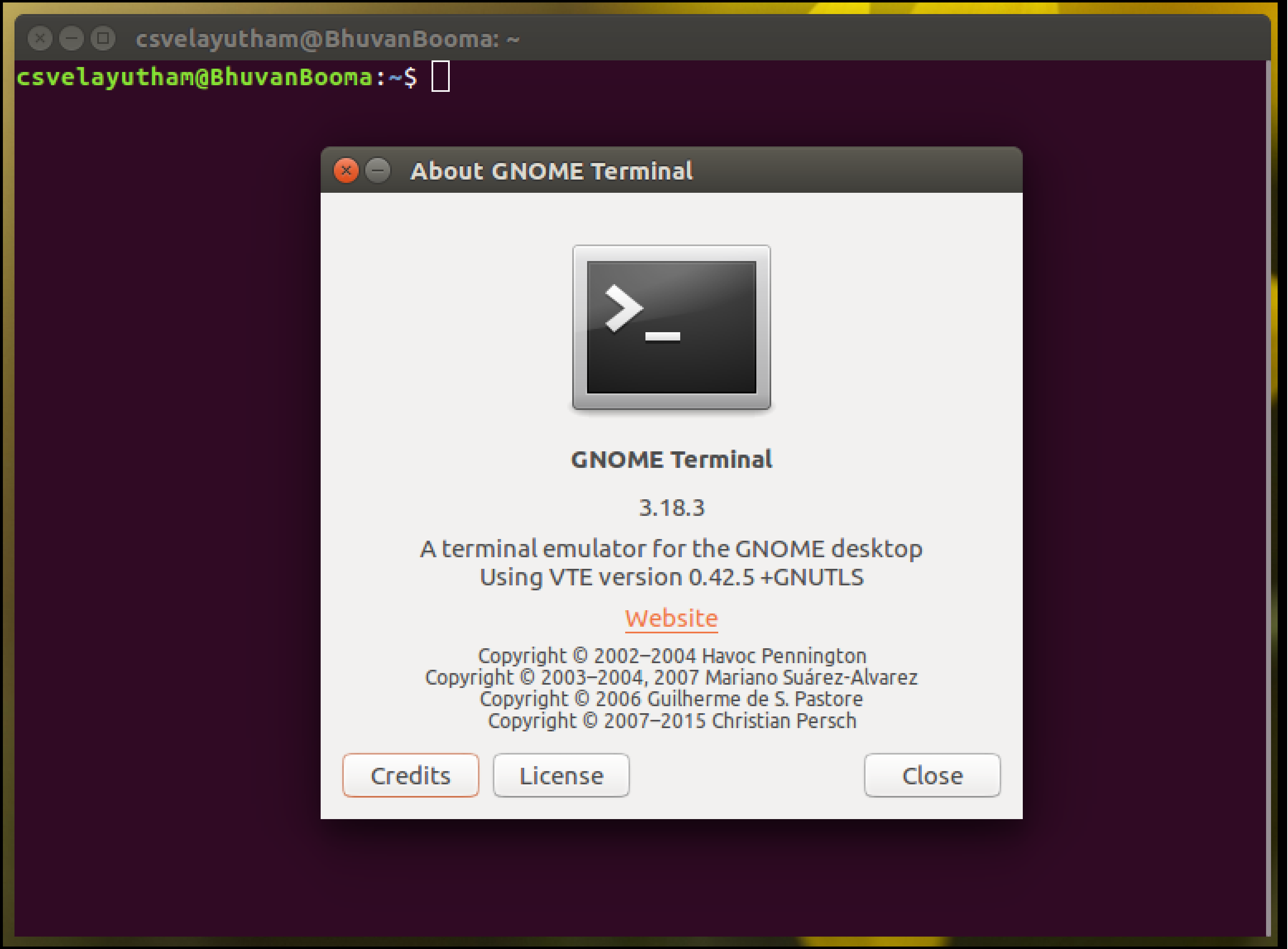
www.gedit.org

Copyright © 1998-2015 - the gedit team

This program comes with absolutely no warranty.
See the GNU General Public License, version 2 or later for details.

Compiler & Terminal

GCC



```
csvelayutham@BhuvanBooma:~$ 
```

GNOME Terminal

3.18.3

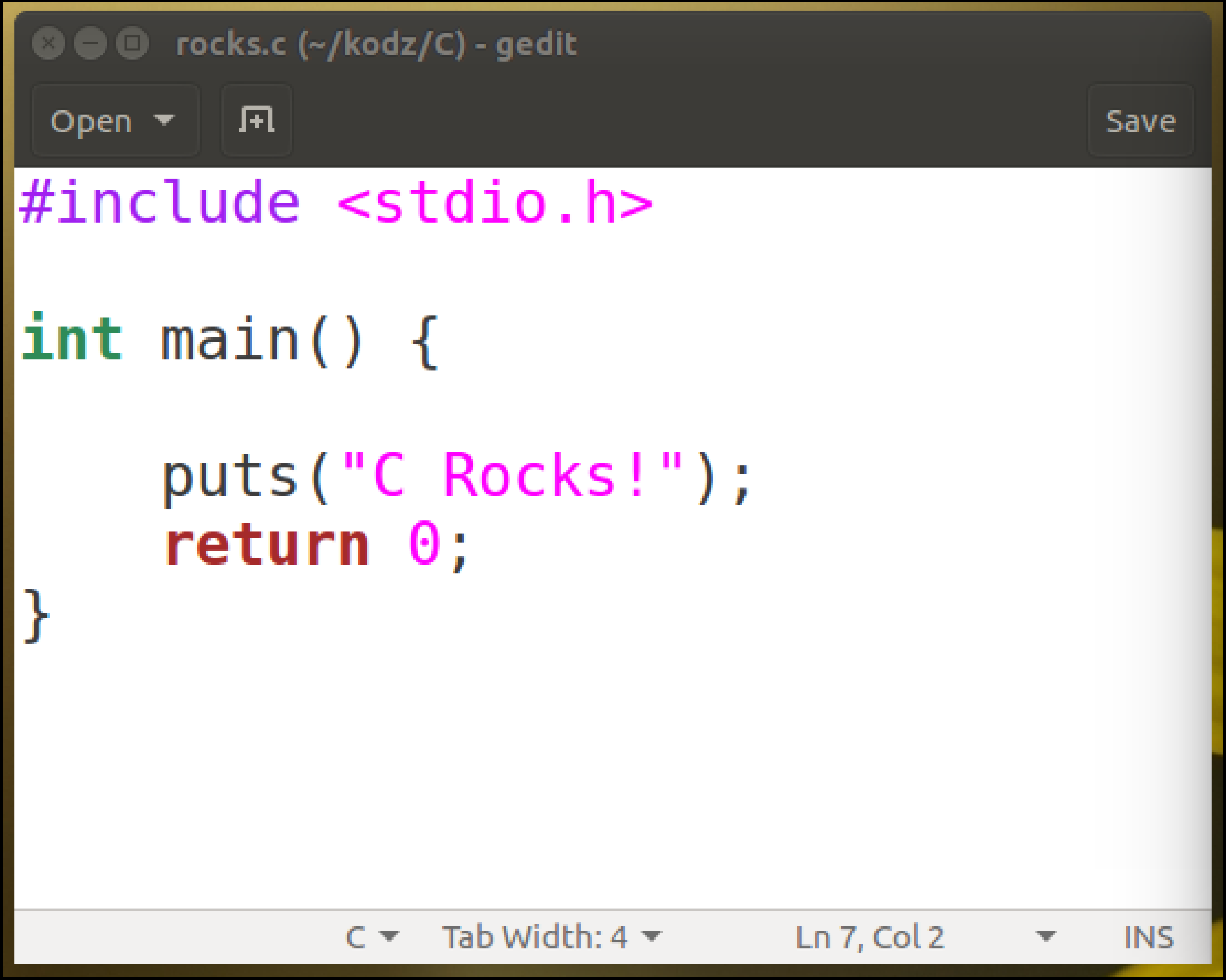
A terminal emulator for the GNOME desktop
Using VTE version 0.42.5 +GNUTLS

Website

Copyright © 2002–2004 Havoc Pennington
Copyright © 2003–2004, 2007 Mariano Suárez-Alvarez
Copyright © 2006 Guilherme de S. Pastore
Copyright © 2007–2015 Christian Persch

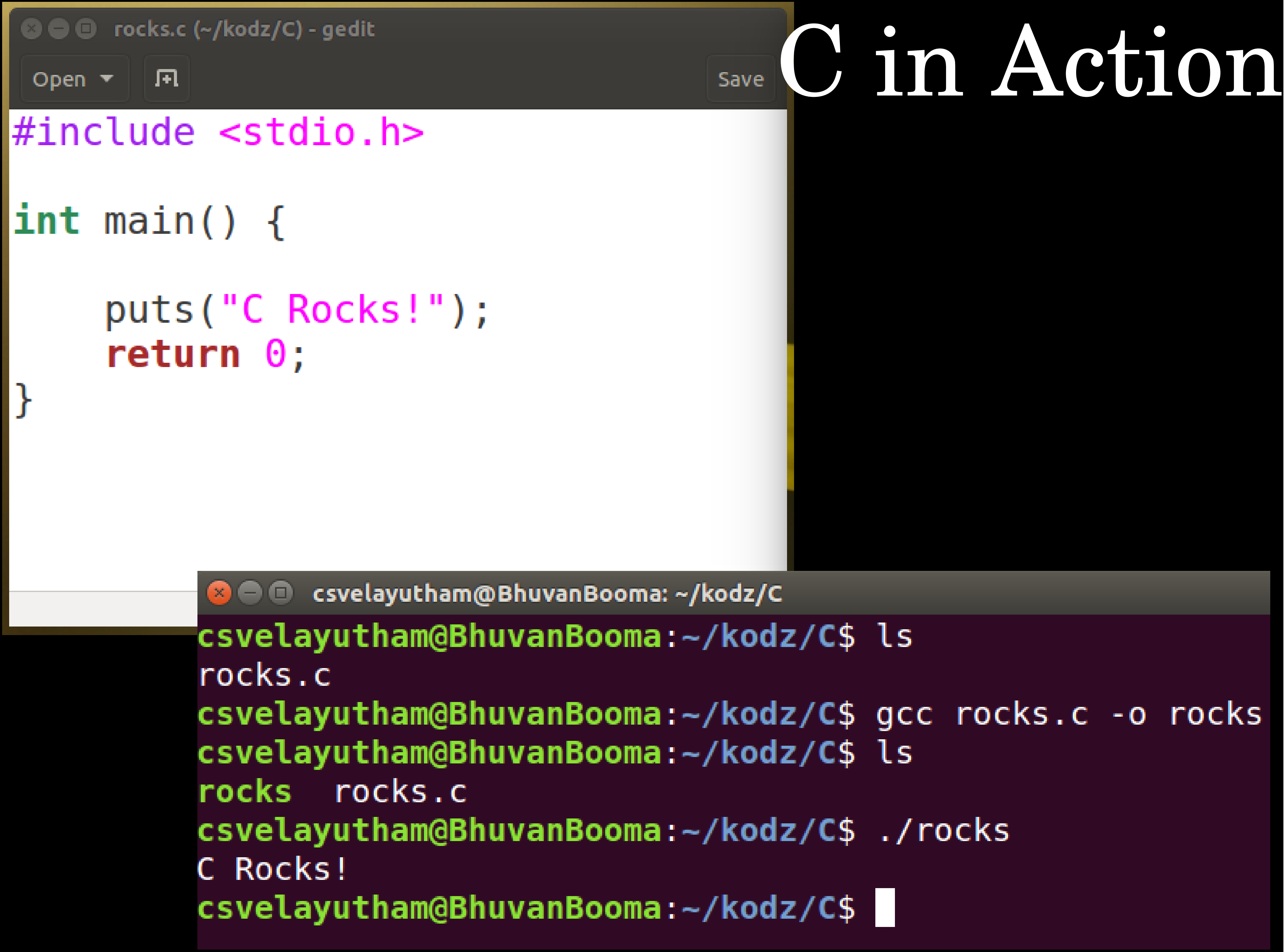
Credits: commons.wikimedia.org

First Program



```c
#include <stdio.h>

int main() {

    puts("C Rocks!");
    return 0;
}
```

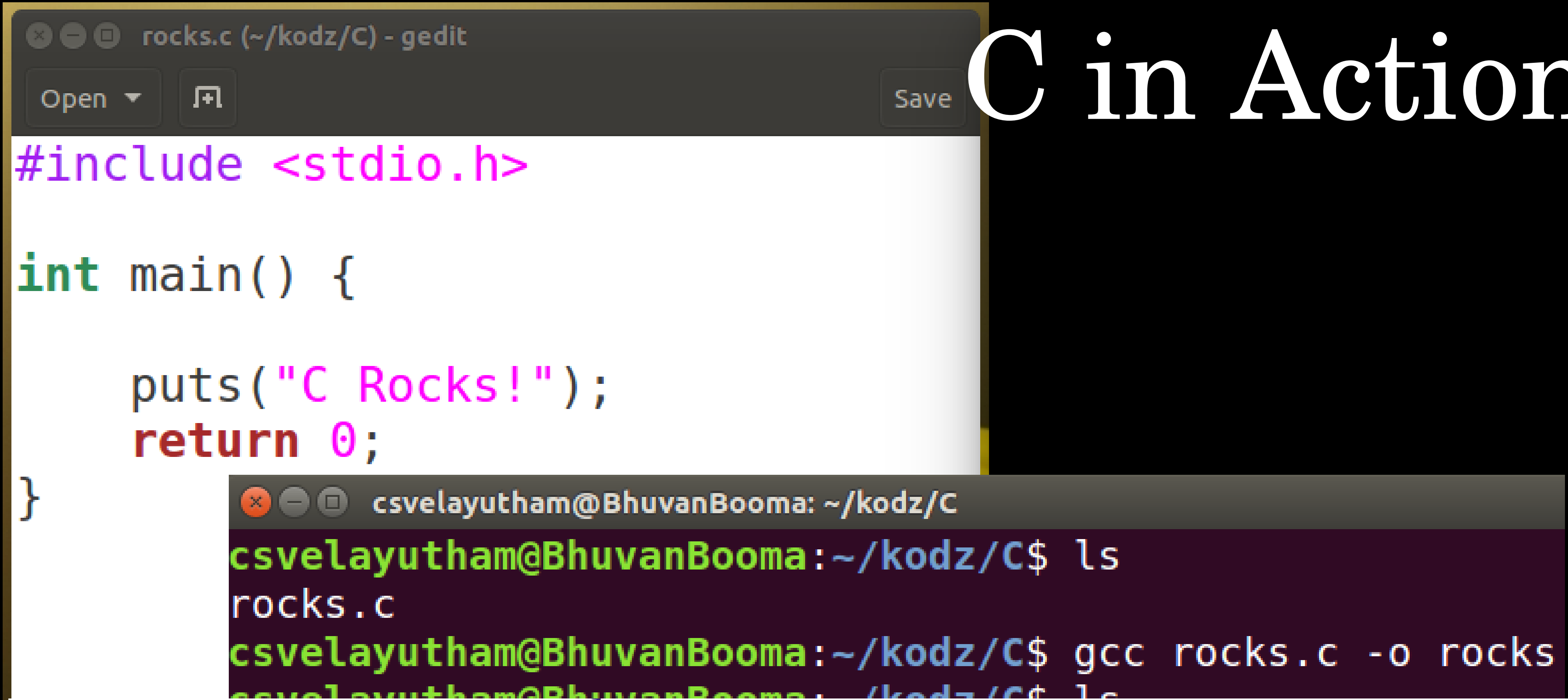
C in Action


rocks.c (~/kodz/C) - gedit
Open
Save
csvelayutham@BhuvanBooma: ~/kodz/C

```c
#include <stdio.h>

int main() {

    puts("C Rocks!");
    return 0;
}
```

```
csvelayutham@BhuvanBooma:~/kodz/C$ ls
rocks.c
csvelayutham@BhuvanBooma:~/kodz/C$ gcc rocks.c -o rocks
csvelayutham@BhuvanBooma:~/kodz/C$ ls
rocks  rocks.c
csvelayutham@BhuvanBooma:~/kodz/C$ ./rocks
C Rocks!
csvelayutham@BhuvanBooma:~/kodz/C$
```

C in Action


rocks.c (~/kodz/C) - gedit
Open
Save
csvelayutham@BhuvanBooma: ~/kodz/C

```c
#include <stdio.h>

int main() {

    puts("C Rocks!");
    return 0;
}
```

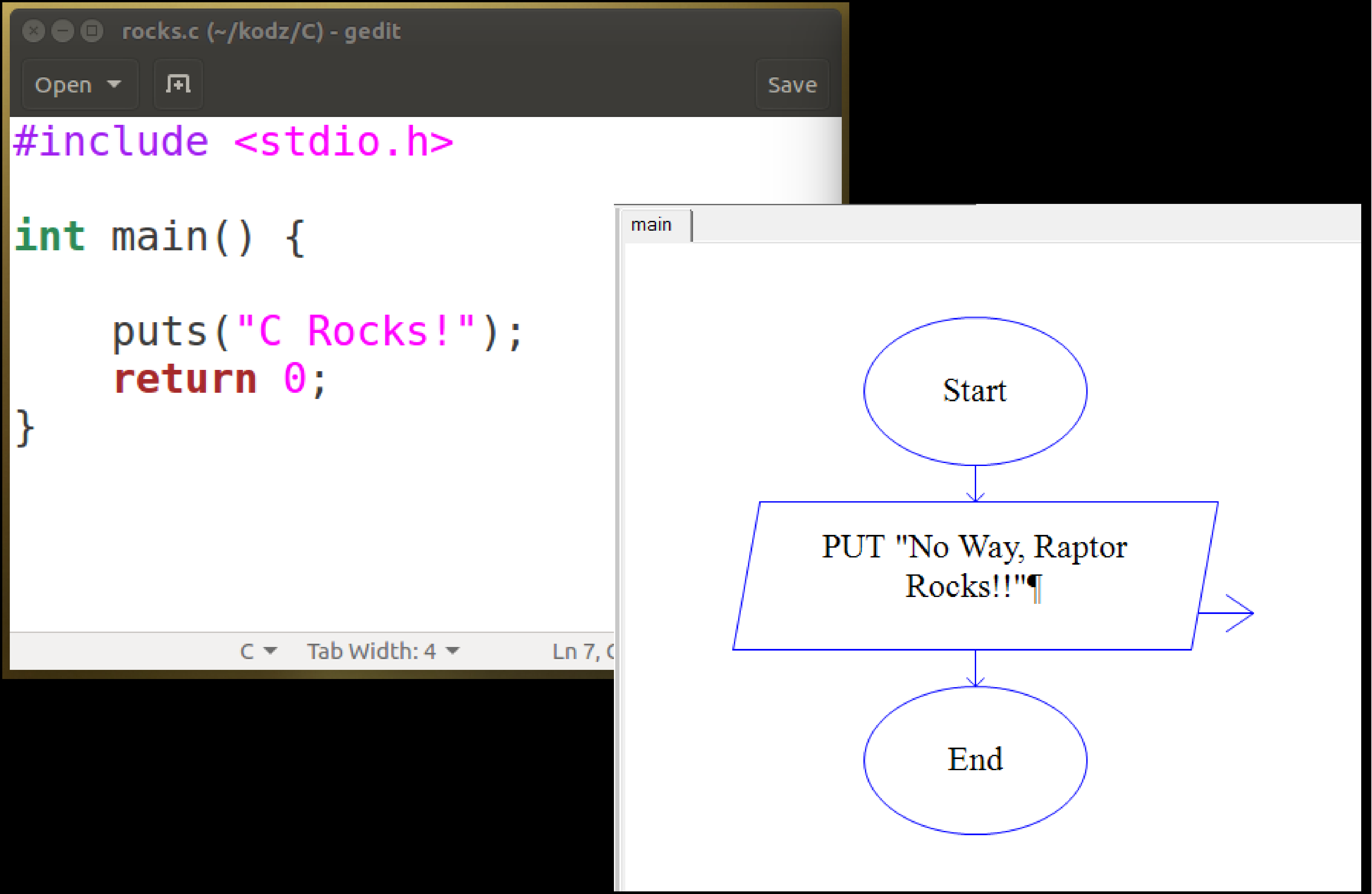
```
csvelayutham@BhuvanBooma:~/kodz/C$ ls
rocks.c
csvelayutham@BhuvanBooma:~/kodz/C$ gcc rocks.c -o rocks
```

there are no Dumb Questions

Q: **Why do I have to prefix the program with `./` when I run it on Linux and the Mac?**

A: On Unix-style operating systems, programs are run only if you specify the directory where they live or if their directory is listed in the PATH environment variable.

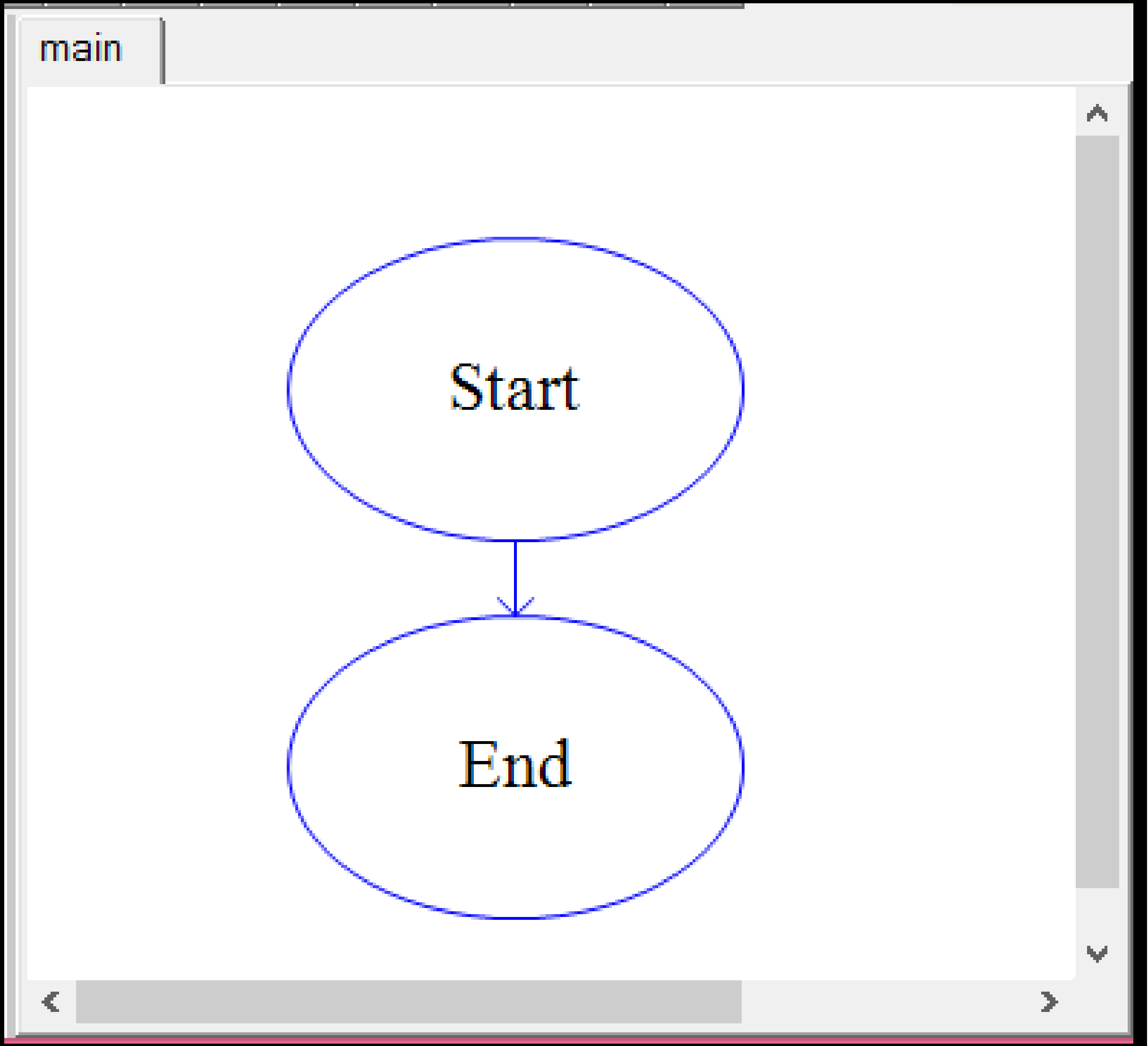
C vs Raptor



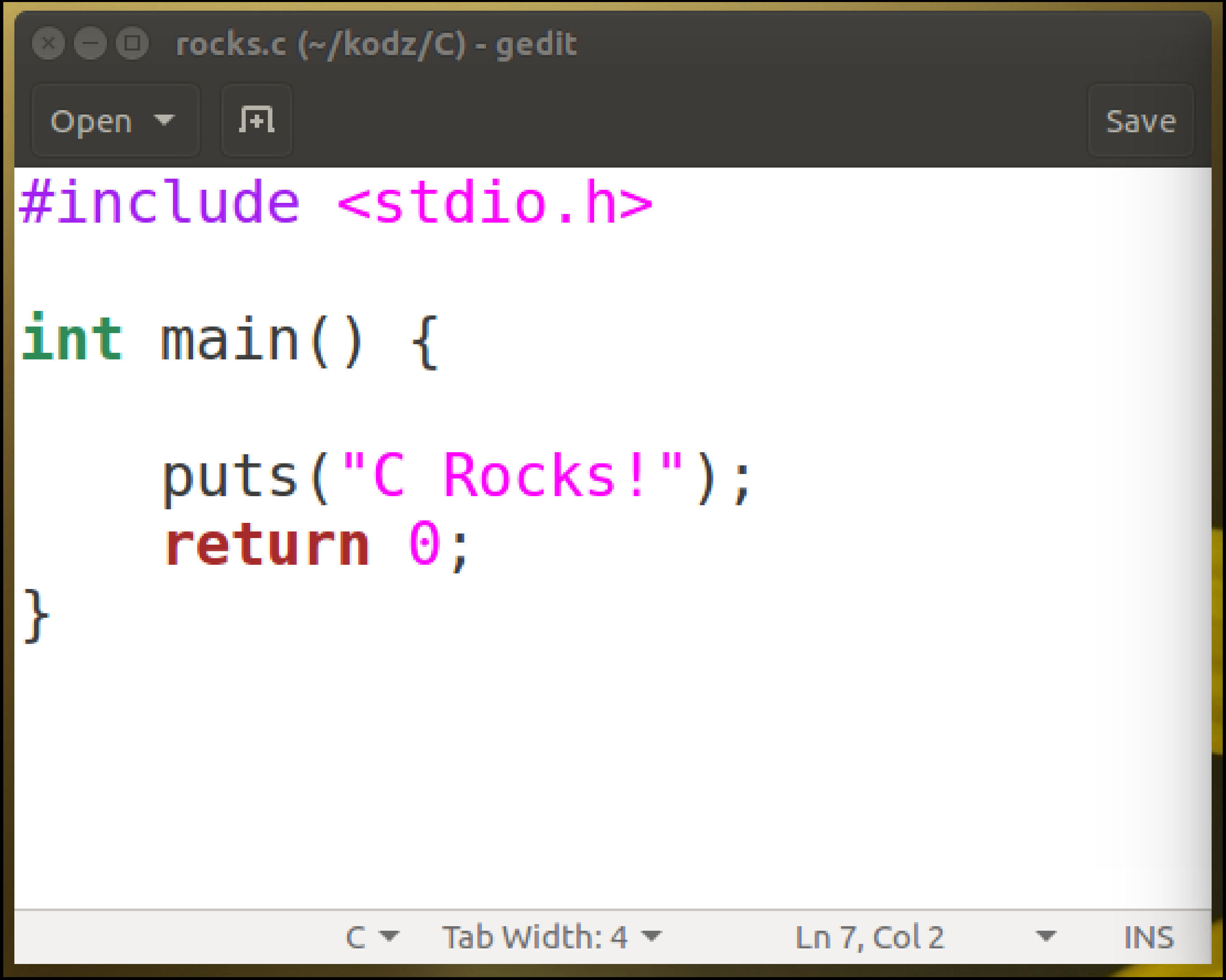
```c
#include <stdio.h>

int main() {

    puts("C Rocks!");
    return 0;
}
```

Barebones

```
int main()
{



   return 0;
}
```



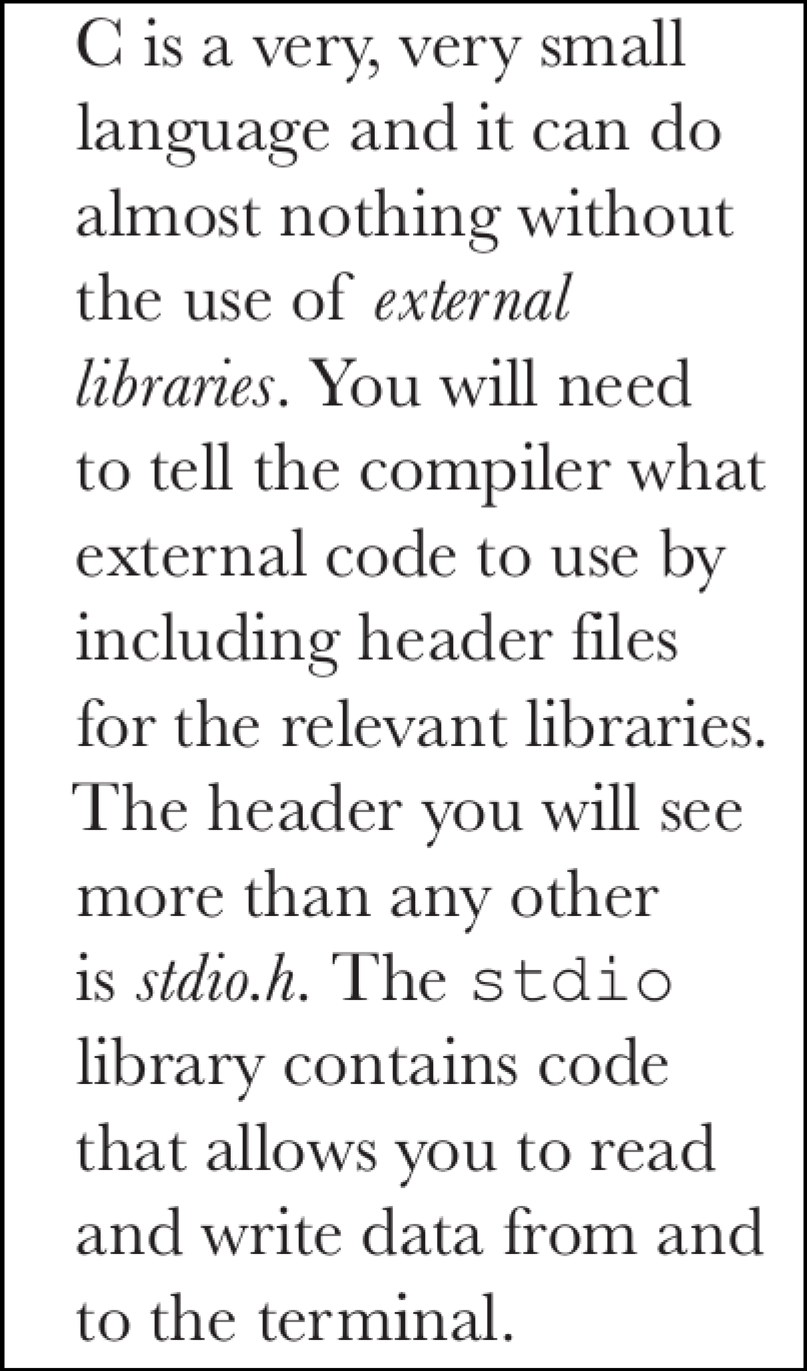
First Program



```c
#include <stdio.h>

int main() {

    puts("C Rocks!");
    return 0;
}
```

Why `#include` ?

```
#include <stdio.h>
```



C is a very, very small language and it can do almost nothing without the use of *external libraries*. You will need to tell the compiler what external code to use by including header files for the relevant libraries. The header you will see more than any other is *stdio.h*. The `stdio` library contains code that allows you to read and write data from and to the terminal.

Credits: Head First C

Anatomy

```
int main()
{
```

Because the function is called "main"
the program always start here
like Raptor remember!!

```
    return 0;
}
```

Anatomy

```
int main()
{
```

The body of (any) function is always surrounded by braces

```
   return 0;
}
```

Anatomy

```
int main()
{

    return 0;
}
```

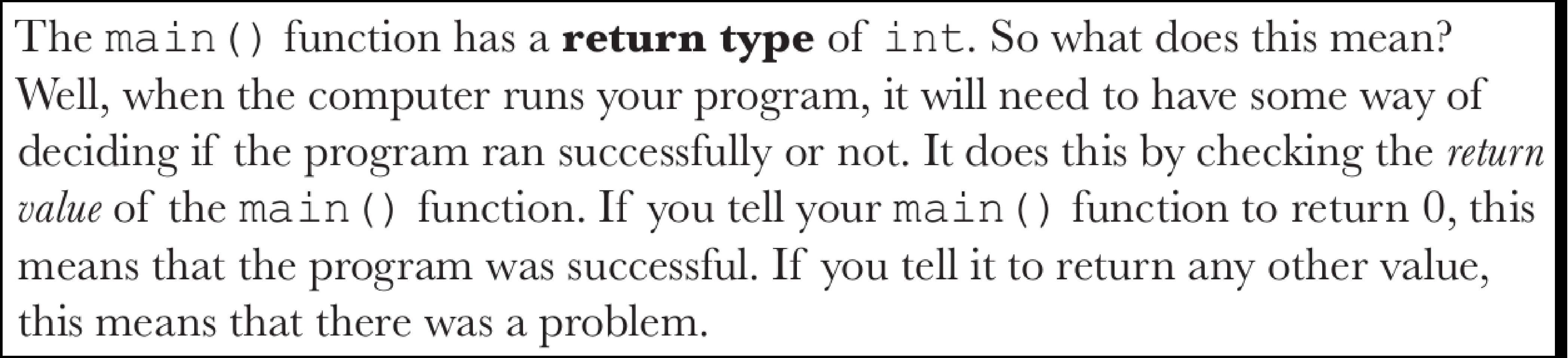
If we have to pass any parameters to main, they'd be mentioned here

Anatomy

```
int main()
{
    return 0;
}
```

This is the return type. It should always be **int** *for the main() function*

there are no Dumb Questions



The `main()` function has a **return type** of `int`. So what does this mean? Well, when the computer runs your program, it will need to have some way of deciding if the program ran successfully or not. It does this by checking the *return value* of the `main()` function. If you tell your `main()` function to return 0, this means that the program was successful. If you tell it to return any other value, this means that there was a problem.

Anatomy

```
#include <stdio.h>
int main()
{
```

Put the following string into
the standard output

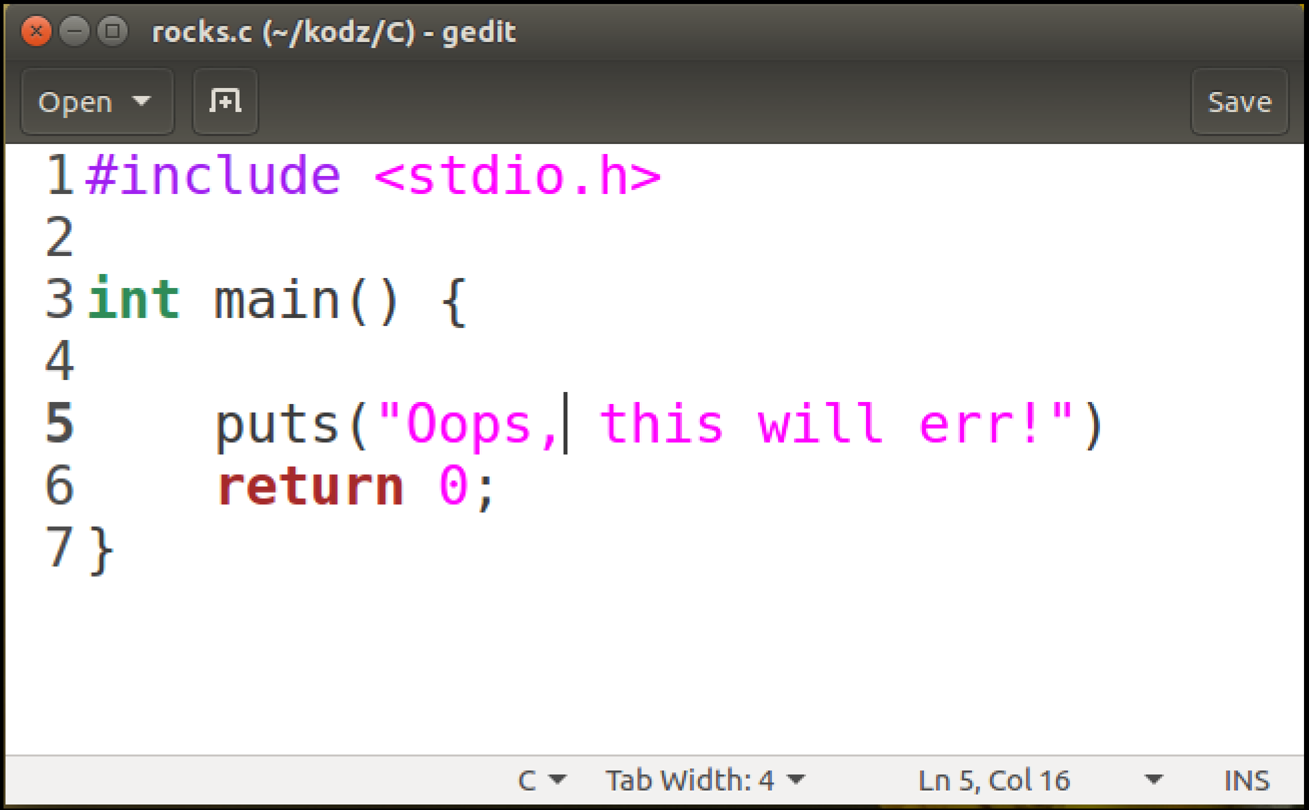
```
    puts(“C Rocks!”);

    return 0;
}
```

Anatomy

```
#include <stdio.h>
int main()
{

    puts(“C Rocks!”);

    return 0;
}
```

The behavior of puts is defined here

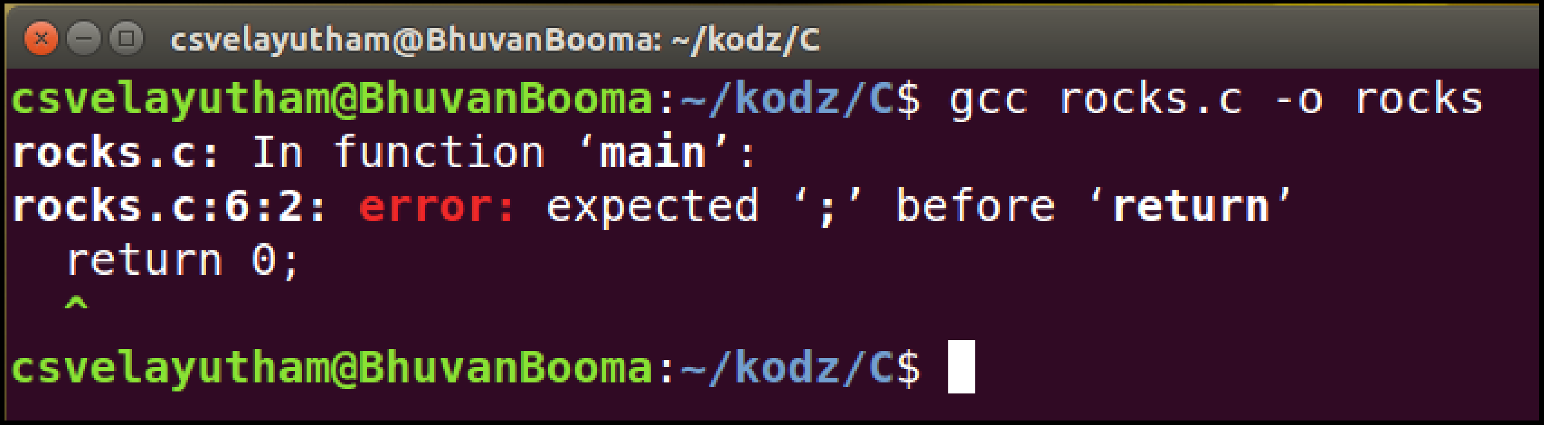
Yikes! Error



```
1 #include <stdio.h>
2
3 int main() {
4
5     puts("Oops, this will err!")
6     return 0;
7 }
```

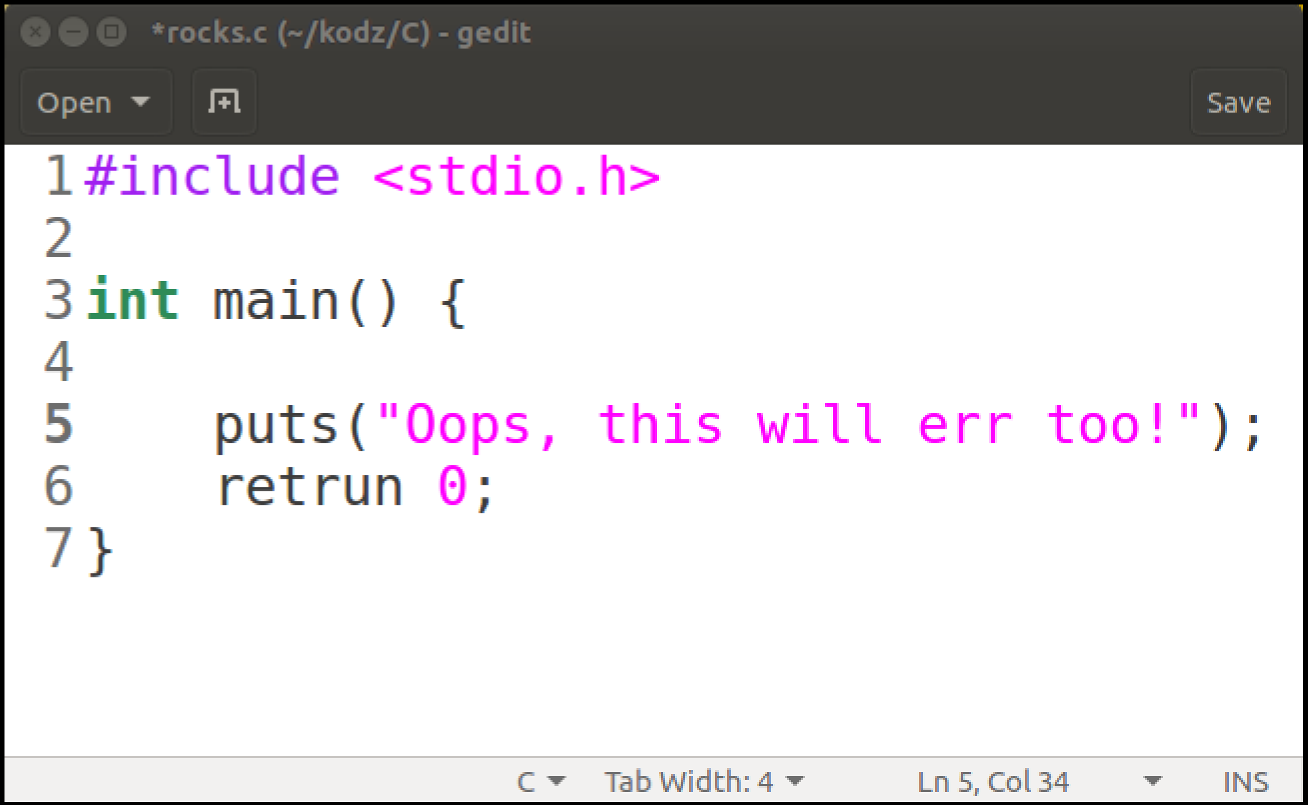
C ▾ Tab Width: 4 ▾ Ln 5, Col 16 ▾ INS

Note the missing Semi-colon



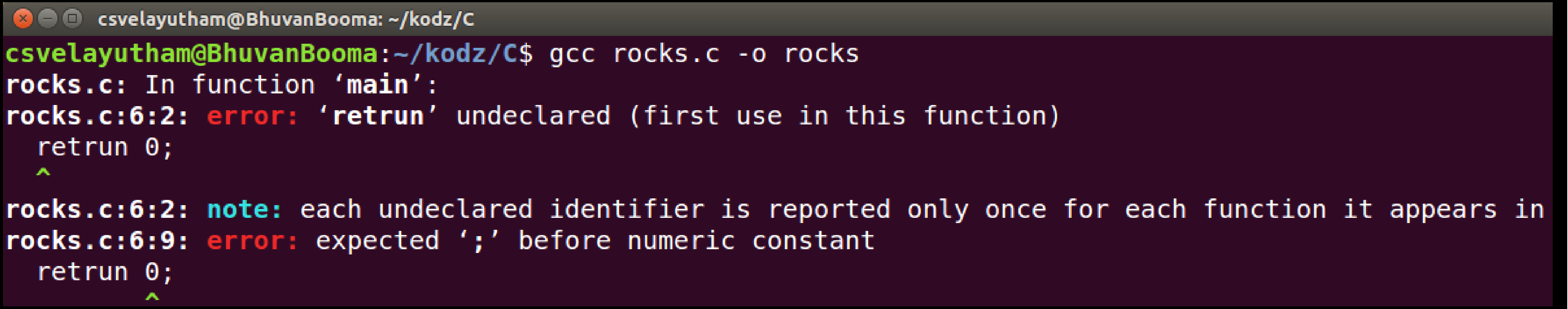
```
csvelayutham@BhuvanBooma:~/kodz/C$ gcc rocks.c -o rocks
rocks.c: In function ‘main’:
rocks.c:6:2: error: expected ‘;’ before ‘return’
  return 0;
  ^
csvelayutham@BhuvanBooma:~/kodz/C$ 
```

Yikes! Error



```c
1 #include <stdio.h>
2
3 int main() {
4
5     puts("Oops, this will err too!");
6     retrun 0;
7 }
```

Note the
typo
retrun
instead of
return!!



```
csvelayutham@BhuvanBooma:~/kodz/C$ gcc rocks.c -o rocks
rocks.c: In function ‘main’:
rocks.c:6:2: error: ‘retrun’ undeclared (first use in this function)
  retrun 0;
  ^
rocks.c:6:2: note: each undeclared identifier is reported only once for each function it appears in
rocks.c:6:9: error: expected ‘;’ before numeric constant
  retrun 0;
         ^
```

Comments

> “Programs must be written for people to read, and only incidentally for machines to execute
>
> – Harold Abelson

15CSE102
Computer Programming

(Next Topic)

Credits: jobssearch24.com