

2nd Edition

AN INTRODUCTION TO

C & GUI

PROGRAMMING

UPDATED
FOR
GTK 3

Simon Long





First published in 2019 by Raspberry Pi Trading Ltd, Maurice Wilkes Building,
St. John's Innovation Park, Cowley Road, Cambridge, CB4 0DS

Publishing Director: Russell Barnes • Editor: Phil King • Sub Editor: Nicola King
Author: Simon Long • Design: Critical Media
CEO: Eben Upton

The publisher, and contributors accept no responsibility in respect of any omissions
or errors relating to goods, products or services referred to or advertised in this book.
Except where otherwise noted, the content of this book is licensed under a Creative
Commons Attribution-NonCommercial-ShareAlike 3.0 Unported
(CC BY-NC-SA 3.0) ISBN: 978-1-912047-45-1. ISSN: 978-1-912047-44-4.



Welcome to **An Introduction to C & GUI Programming**

The C programming language was invented in the early 1970s, and since then has become one of the most popular and widely used general-purpose languages. C can be used to create simple command-line programs, or embedded code to operate the tiny microcontrollers in toasters and watches. At the other extreme, it can be used to create rich graphical desktop applications – in fact, most of Linux (and Raspberry Pi OS itself) is written in it. It can give you control over the smallest details of how a processor operates, but is still simple to learn and read. The first part of this book is an introduction to programming in C for absolute beginners; the second part shows how to use C to create desktop applications for Raspberry Pi OS. You don't need any programming experience, and a Raspberry Pi running Raspberry Pi OS is all you need to get started.

About the Author

Simon Long is an engineer working for Raspberry Pi. He is responsible for the Raspberry Pi Desktop and its associated applications. Before joining Raspberry Pi, he worked for Broadcom, where he first met Eben Upton, and before that spent ten years working as a software engineer and user interface designer for a major consultancy firm. In his spare time, he enjoys solving those really hard crosswords without any black squares.



"Dedicated to the memory of Pythagoras, a very special cat, and my devoted companion while writing this book."

Contents

Chapter 1: Getting started	010
Learn how to use C to program the Raspberry Pi	
Chapter 2: Variables and arithmetic	014
Create variables and do maths	
Chapter 3: Conditions and comparisons	019
Control the flow of your C programs	
Chapter 4: More advanced flow control	024
For loops and case statements	
Chapter 5: Pointers	030
Variables have addresses too	
Chapter 6: Functions	035
Split your code into bite-sized chunks	
Chapter 7: Arrays and strings	041
Handle lists of values, and letters	
Chapter 8: The string library	046
Simplify common operations on strings	
Chapter 9: User Input	052
Reading and interpreting user input	
Chapter 10: File input and output	058
Learn to read from and write to files	

Chapter 11: More about types and variables	063
Type definitions, enumerations, and more	
Chapter 12: Header files and the preprocessor	068
Splitting code up into multiple files	
Chapter 13: Introduction to GTK	074
Get ready to start creating GUI elements	
Chapter 14: Your first GTK program	077
Start coding in C with the GTK library	
Chapter 15: Buttons	081
Make your window do something by adding a button	
Chapter 16: Labels and layout	085
Use a box widget to add a text label to your window	
Chapter 17: More advanced layout	091
Expand your window, and position and resize buttons automatically	
Chapter 18: User input	097
Enable users to enter text and data, and to select options	
Chapter 19: Combo boxes and list stores	103
Create combo boxes for user input and associate list stores with them	
Chapter 20: Tree views	109
Use the GtkTreeView widget to display information	
Chapter 21: Menus	115
Create menu bars with drop-down menus	
Chapter 22: Dialogs	120
Give users information and ask them questions	

Chapter 23: Built-in dialogs	123
GTK contains some ready-made dialogs	
Chapter 24: Customising widgets	131
Change the properties of widgets	
Chapter 25: Glade	135
Create window layouts for applications	
Chapter 26: Quick reference	144
Cheat sheets and code examples	

Chapter 1

Getting started

C is one of the most widely used programming languages
– learn how to use it to program the Raspberry Pi!

What's so great about C?

C is a very versatile and widely used programming language. It has been used to write pretty much everything, from low-level routines to control the hardware in embedded microcontrollers to complete operating systems like Linux with graphical user interfaces. In spite of this huge flexibility, it is also relatively simple – the language only has about 20 or so keywords, but there are huge libraries of additional functions that you can call on when you need them. In the first part of this book, we are going to concentrate on learning about the keywords, with a few of the more commonly used library functions; the second part of the book shows how to use the GTK library to make it easy to write graphical interfaces in C.

Many of the languages that you may have seen, such as Python, are what are called *interpreted languages*. This means that the code you write is run directly: each line of code is read in and interpreted as you run it. C is different: it's a *compiled language*. This means that the code you write, known as the *source code*, is never run directly. The source code is passed through a program called a *compiler*, which converts it into a machine-readable version called an *executable* or a *binary*; you then run the resulting executable.

This may seem complex, but it has a few big advantages. First, it means that you don't need to have a copy of C itself on every computer you want to run your program on; once compiled, the executable is standalone and self-contained. Second, the compilation process will find a lot of errors before you even run the program (but it won't usually find all of them). Most importantly, the compilation process means that the time-consuming translation of human-readable code into machine-readable instructions has already happened, which means that compiled code generally runs many times faster than interpreted code would.

CHOOSE YOUR EDITOR

You can use whatever editor you like to enter code, as long as it saves it as plain text. The Geany editor included in Raspberry Pi OS is a good choice, but you can also use Leafpad, nano, or any others that you prefer.