|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Baking Pi – Operating Systems Development  **This course has not yet been updated to work with the Raspberry Pi models B+ and A+. Some elements may not work, in particular the first few lessons about the LED. It has also not been updated for Raspberry Pi v2.**  Welcome to Baking Pi: Operating Systems Development! Course by [Alex Chadwick](mailto:awc32@cam.ac.uk).  You can now help contribute to this tutorial on [GitHub](https://github.com/chadderz121/bakingpi-www).  This website is here to guide you through the process of developing *very* basic operating systems on the [Raspberry Pi](http://www.raspberrypi.org/)! This website is aimed at people aged 16 and upwards, although younger readers may still find some of it accessible, particularly with assistance. More lessons may be added to this course in time.  This course takes you through the basics of operating systems development in assembly code. I have tried not to assume any prior knowledge of operating systems development or assembly code. It may be helpful to have some programming experience, but the course should be accessible without. The [Raspberry Pi forums](http://www.raspberrypi.org/phpBB3/viewforum.php?f=72) are full of friendly people ready to help you out if you run into trouble. This course is divided into a series of 'lessons' designed to be taken in order as below. Each 'lesson' includes some theory, and also a practical exercise, complete with a full answer.  Rather than leading the reader through the full details of creating an Operating System, these tutorials focus on achieving a few common tasks separately. Hopefully, by the end, the reader should know enough about Operating Systems that they could try to put together everything they've learned and make one. Although the lessons are generally focused on creating very specific things, there is plenty of room to play with what you learn. Perhaps, after reading the lesson on functions, you imagine a better style of assembly code. Perhaps after the lessons on graphics you imagine a 3D operating system. Since this is an Operating Systems course, you will have the power to design things how you like. If you have an idea, try it! Computer Science is still a young subject, and so there is plenty left to discover!   |  | | --- | | **Contents**   * [1 Requirements](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/index.html#requirements)   + [1.1 Hardware](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/index.html#hardware)   + [1.2 Software](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/index.html#software) * [2 Lessons](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/index.html#lessons) |   1 Requirements  **1.1 Hardware**  In order to complete this course you will need a Raspberry Pi with an SD card and power supply. It is helpful, but not necessary, for your Raspberry Pi to be able to be connected to a screen and keyboard.  In addition to the Raspberry Pi used to test and run your operating system code, you also need a seperate computer running Linux, Microsoft Windows or Mac OS X capable of writing to the type of SD card used by your Raspberry Pi. This other computer is your development and support system.  **1.2 Software**  In terms of software, you require a GNU compiler toolchain that targets ARMv6 processors. You will install or build a set of tools, called a cross-compiler, on your development system. This cross-compiler converts your source code files into Raspberry Pi-compatible executable files which are placed on the SD card. The SD card is then transferred to the Raspberry Pi where the executable can be tested.  You can find instruction for getting the toolchain on the [Downloads Page](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/downloads.html), along with model answers for all of the exercises.  2 Lessons   | Table 2.1 - Lessons | | | | --- | --- | --- | |  | **Name** | **Description** | | 0 | [Introduction](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/introduction.html) | This introductory lesson does not contain a practical element, but exists to explain the basic concepts of what is an operating system, what is assembly code, and other important basics. If you just want to get straight into practicals, it should be safe to skip this lesson. | | **OK LED Series (Beginner)** | | | | 1 | [OK01](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/ok01.html) | The OK01 lesson contains an explanation about how to get started and teaches how to enable the 'OK' or 'ACT' LED on the Raspberry Pi board near the RCA and USB ports. | | 2 | [OK02](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/ok02.html) | The OK02 lesson builds on OK01, by causing the 'OK' or 'ACT' LED to turn on and off repeatedly. | | 3 | [OK03](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/ok03.html) | The OK03 lesson builds on OK02 by teaching how to use functions in assembly to make more reusable and rereadable code. | | 4 | [OK04](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/ok04.html) | The OK04 lesson builds on OK03 by teaching how to use the timer to flash the 'OK' or 'ACT' LED at precise intervals. | | 5 | [OK05](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/ok05.html) | The OK05 lesson builds on OK04 using it to flash the SOS morse code pattern (...---...). | | **Screen Series (Advanced)** | | | | 6 | [Screen01](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/screen01.html) | The Screen01 lesson teaches some basic theory about graphics, and then applies it to display a gradient pattern to the screen or TV. | | 7 | [Screen02](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/screen02.html) | The Screen02 lesson builds on Screen01, by teaching how to draw lines and also a small feature on generating pseudo random numbers. | | 8 | [Screen03](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/screen03.html) | The Screen03 lesson builds on Screen02 by teaching how to draw text to the screen, and introduces the concept of the kernel command line. | | 9 | [Screen04](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/screen04.html) | The Screen04 lesson builds on Screen03 by teaching how to manipulate text to display computed values on the screen. | | **Input Series (Advanced)** | | | | 10 | [Input01](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/input01.html) | The Input01 lesson teaches some theory about drivers, and linking programs, as well as keyboards. It is then applied to print out input characters to the screen. | | 11 | [Input02](http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/input02.html) | The Input02 lesson builds on Input01 by teaching how to make a command line interface for an Operating System. | |  |