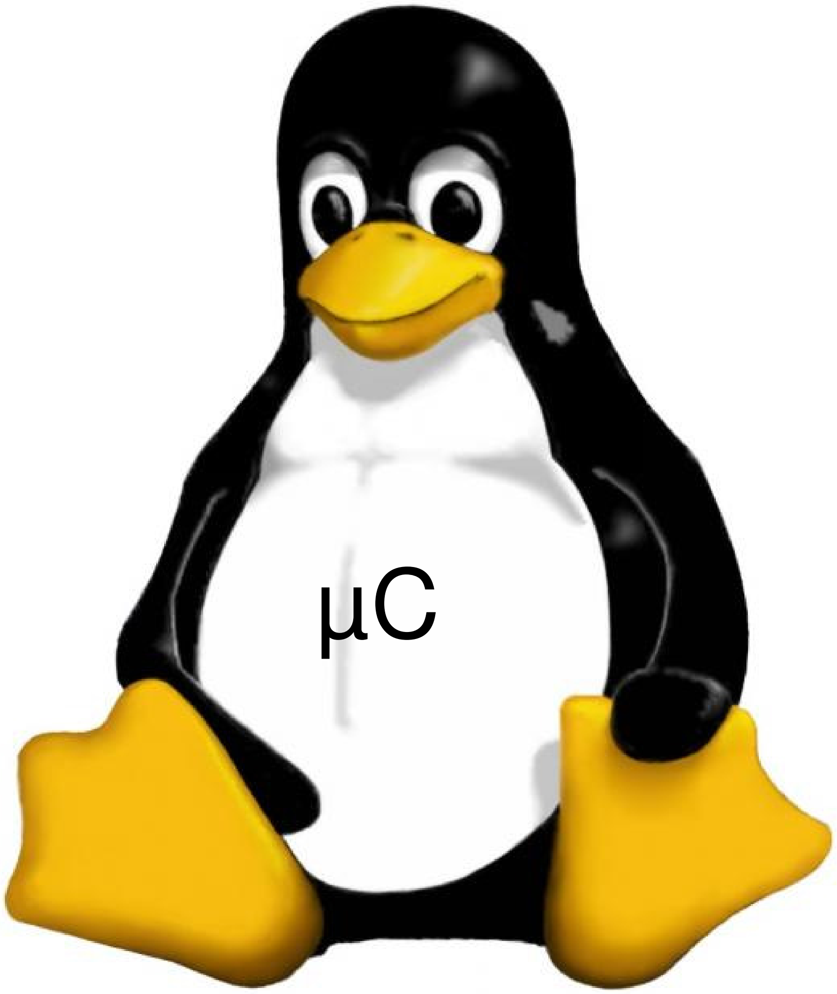
Getting Started With µClinux Development



Embedded Artists

Embedded Artists AB
Södra Promenaden 51
SE-211 38 Malmö
Sweden

info@EmbeddedArtists.com
http://www.EmbeddedArtists.com

Disclaimer

Embedded Artists AB makes no representation or warranties with respect to the contents hereof and specifically disclaims any implied warranties or merchantability or fitness for any particular purpose. Information in this publication is subject to change without notice and does not represent a commitment on the part of Embedded Artists AB.

Feedback

We appreciate any feedback you may have for improvements on this document. Please send your comments to support@EmbeddedArtists.com.

Trademarks

All brand and product names mentioned herein are trademarks, services marks, registered trademarks, or registered service marks of their respective owners and should be treated as such.

How to get this book?

Information about the book

Thank you for viewing this preview version. The full version is a comprehensive 168 pages, covering all the topics included in the Table of Contents.

The book is designed to help you get acquainted with µClinux, the Linux distribution for processors without a Memory Management Unit (MMU). It has a practical approach with lots of step-by-step guides. The guides have been designed around the Embedded Artists LPC2468 OEM Board and LPC2478 OEM Board with appropriate Base Board and for the Embedded Artists µClinux port for the LPC24xx microcontroller.

Free of charge with purchase of our kits

You can download this book from our support pages free with the purchase of our LPC2468 Developer's Kit http://www.embeddedartists.com/products/kits/lpc2468_kit.php or our LPC2478 Developer's Kit http://www.embeddedartists.com/products/kits/lpc2478_kit.php.

Table of Contents

1 Introduction

This book is designed to help you get acquainted with µClinux, the Linux distribution for processors without a Memory Management Unit (MMU). The book has a practical approach with lots of step-by-step guides. The guides have been designed around the Embedded Artists LPC2468 OEM Board and LPC2478 OEM Board with appropriate Base Board and for the Embedded Artists µClinux port for the LPC24xx microcontroller.

The development environment used for the exercises is based on a Debian Etch Linux distribution which is distributed as a VMware image that can be run in, for example, the VMware Player available for Windows PCs as well as Linux PCs, i.e., a virtualization approach has been chosen for the Debian Etch distribution.

It is not necessary to have expert knowledge about using Linux in order to understand the content of this book or to do the exercises since one part of the book will cover the basics of using Linux.

Besides describing µClinux the Universal boot loader (U-boot) will also be covered in this book since without a boot loader it will be difficult to get µClinux up-and-running. There should be enough information in this book to get you working with µClinux on an Embedded System.

1.1 What is an Embedded System?

The term Embedded Systems is generally defined to mean a computer system designed to solve one or a very few specific functions. These functions may need to be performed during long periods of time without interruption or even interaction with a person. Because of this they must in general be reliable and stable, maybe even meet real-time and safety critical requirements. It is not acceptable to have to reboot an embedded system every day or even on a weekly or monthly basis, they may have to be able to run continually for several years. The computer system is *embedded* in a sense where it is put into a device in a way where the device is not perceived as being a computer system.

This term is quite general and applicable for a lot of different devices found in the everyday world today. There are in fact a lot more embedded systems around you than you would probably imagine, ranging from simple sensors such as a thermostat, thermometer or motion sensor to a TV, washing machine, mobile phone or parts in a modern car or airplane. A modern car today can have 60 or more embedded computers controlling everything from fuel injection, the power windows, airbags and brakes.

The opposite of an Embedded System is a general purpose computer system that can be used to perform many types of tasks and run many types of applications. The Personal Computer (PC) found in many homes today is a general purpose computer system. It can be used for word processing, photo editing, software development, web browsing, entertainment (play games, listen to music, watch movies), heavy computational tasks and much more.

The mobile phone is on the list of embedded devices, but the question is if this is still true. When the mobile phone could only be used to make a phone call and maybe send and receive text messages the definition would apply for a mobile phone, but the phones on the market today can be used for a lot more; take photographs, play music, browse the web, send and receive e-mails, navigation (using GPS), and lots more, i.e., it is more of a general purpose computer system. This last paragraph just want to point out that sometimes it might be difficult to say if a device is really considered to be an embedded system in the true original meaning of the term. Nevertheless it is a device with a computer system probably in the need for an operating system.

1.2 The Operating System

Why would an Operating System (OS) be needed in an embedded system? First of all it is not sure it would be needed at all. This depends on the situation and what the embedded system is supposed to do and which problem it must solve. For the simplest device only performing one task such as regularly reading a sensor value, an operating system would probably not be needed, while a more complex device where several sensors should be read, a display regularly updated, and data sent onto a wireless network an operating system would probably be a necessity.

The responsibility of the operating system is to manage the resources, such as processing power (access to the processor), memory, and other input- and output devices attached to the computer system. It lies as a layer between the applications and the actual hardware making sure access is handled in a fair and controlled way. Many different types of operating systems exist and below is a list of a couple of different types to know about.

- **Multitasking or single-tasking OS** – In a multitasking OS multiple tasks (also called processes) can be performed simultaneously although the computer system only has one CPU. It is the operating systems responsibility to divide the access time to the CPU into smaller parts, time slices, and to schedule the processes so that they are given one or more time slices at a time. How much time that is given to a process is dependent on a specific scheduling algorithm.

  In a single tasking OS several processes may exist, but only one will run at any given time and the next process will not start to run until the first has stopped executing.
- **Real-Time Operating System (RTOS)** – This is an operating system usually found in embedded systems. The applications running in this OS need to react on input and deliver an output in real-time, sometimes with requirements on guaranteeing that deadlines are met and that the behaviour is deterministic. The RTOS is usually a multitasking operating system allowing several applications to run simultaneously.
- **Multi-user or single-user** – As the name implies a multi-user operating system allows for several users to have concurrent access to the computer while a single-user system only allows one user access. A server is typically running a multi-user operating system while a mobile phone usually only needs a single-user operating system.

1.3 Choosing Linux

In the annual survey conducted in the year 2007 by Embedded System Design, see ref [1], the participants were asked which operating system *type* they would be likely to use in their next embedded project. The majority would chose a commercial OS, but the number of developers choosing an open-source OS is high; 27 % as shown in Figure 1. Looking at the trend it reveals that the number of developers choosing a commercial OS is dropping.

According to the same survey 21 % of the participants were already using Linux as the open-source OS in their embedded project and 31 % were likely to use it in the next 6 to 12 months. The reason for choosing Linux was mostly because of low cost, adaptability/extensibility of the OS and personal control of its features and migration.

The survey shows that Linux, a multi-user and multitasking operating system, has become popular to use also in embedded systems, not only desktops and servers and that low cost is the prominent factor when choosing an open-source operating system for an embedded project.