Embedded Linux for Distributed Networks

People know Linux
Please talk about embedded Linux.

Abstract

The following paper introduces youwhom? to the world of Embedded Linux and its applications in the area of Distributed Networks. The paper discusses why Linux is well suited for embedded applications and the advantages it holds over the other operating systems. The paper is organized into three parts. In the first part, an overview of Linux and the reasons for embedding it are discussed. The next part highlights the aspects of embedding Linux. The paper concludes by discussing some applications where embedded Linux is used. //too little//

1. Introduction

1.1 Overview of Linux

Linux is an open source Unix-like kernel that can be freely distributed under the terms of GNU General Public License. It was developed by a Finnish student Linus Torvalds and was released officially in 1991. Linux is a multi-tasking, multi-user, multi-processor operating system supporting a wide range of hardware platforms like X86, Alpha, Sparc etc. One of the great strengths of the which?operating system is the support for networking. This includes a large number of drivers for network face adaptors and a complete implementation of the TCP/IP networking. There are also drivers available for USB devices, multimedia devices and many other types of hardware[1].

1.2 Problems with the traditional Operating Systems [2]

- Stand-alone code: Too simplistic for today's technology and functionality requirements.
- Plain old DOS: Out of touch with today's technology and functionality requirements.
- Windows: Perceived as unreliable, fat and expensive.
- Unix: Not particularly embeddable; expensive.
- Proprietary RTOSes: Lack of standards, expensive development and license costs.

1.3 The whys of Embedded Linux

Linux was developed specifically as an operating system for the desktop/server environment. More recently, there has been a growing interest in tailoring Linux to the different hardware and software needs of the embedded applications environment. In hardware terms, the minimum requirements of desktop Linux compare favorably with operating systems such as Windows and Solaris. Red Hat, a Linux distributor, suggests a minimum installation of an Intel 386 above for the CPU, a 1620 Mbyte hard disk for a server, or 450 Mbytes for a workstation, and 16 Mbytes of RAM. An embedded operating system must also display a degree of robustness well beyond the requirements of the desktop domain. A Linux server, or even a desktop machine, expects a specific power- down sequence to occur before the power supply is removed from the system. If the power fails unexpectedly a system can become corrupt. Since power cycling and power failures are more likely in embedded environments, the system is required to handle these without user intervention[1].

So in a nutshell the considerations that make Linux attractive to the embedded world are enumerated below

- *No licensing fee is needed*—Since Linux is distributed free under the terms of GNU the users don't have to pay for using it.
- *Open Source*—The kernel and support applications are open-source, meaning that the users have direct access to the source code.
- *Reliability*—Linux has a good reputation for its robustness. Up-time is measured in weeks or years and systems rarely require rebooting.
- *Scalability*--With the advent of Linux 2.4, the kernel has become significantly more modular and scalable. This is a distinct advantage when using Linux in resource-constrained environments.
- *Support*—Coping up with a new operating system can be quite problematic. In the case of Linux, extensive documentation is available from the Internet, and companies such as Red Hat provide support, training, seminars and courses.
- *Portability*—It is very easy to develop/debug an application on a host system running Linux and then transfers to the target, thus reducing the time for debugging. [1]

2. Implementing Embedded Linux

A minimal embedded Linux system needs the following components to function properly:

The Initial Program Loader:

Before the Linux kernel can start running it has to be loaded into the memory. When running from standard devices like hard disk drive, standard IPLs such as LILO are readily available. But when running from ROM or from flash EPROM, which is the likely setup in an embedded system, a hardware specific loader has to be provided [1].

The kernel:

The kernel is the heart of the Linux operating system. Linux has a monolithic kernel meaning that the whole operating system consisting of process management, memory management, file system and drivers is contained within one binary image. Since Linux is open source and modular, a user can recompile the kernel to obtain the functionality required. A compiled kernel is generally stored in a compressed form and is then uncompressed when it has been loaded into memory. A compressed kernel is typically in the range of 400-740 Kbytes depending on the degree of functionality supported. A benefit of the kernel architecture is support for 'loadable modules', facilitating the extension of the kernel by linking additional code at runtime. Loadable modules allow the user to add support to the kernel for additional devices at any time. They are also extremely useful during development as they can be loaded and unloaded on the fly. Application-specific requirements may necessitate the modification or 'patching' of the kernel. Real-time support would be a prime example of an instance where kernel modification would be required. [1]

Components of the File System:

The Linux kernel mounts a 'root' file system when booted. Depending on the required configuration, this may be located on a floppy or hard disc drive, a ROM or flash memory, or on a share of a drive accessed over a network. The file system must contain a minimal set of devices, libraries, utilities, configuration files and scripts to get the system up and running. The

majority of the file system is taken up by system utilities and their associated shared library files. There are at least two open-source projects that are useful when considering which applications and utilities to place on the file system: 'BusyBox' which encapsulates around 125 of the most common Linux commands in one binary executable, and 'TinyLogin' which supports nine of the commonly required login and authentication commands. Both of these projects are designed to be highly modular [1].

Linux applications are also linked to shared libraries. These are libraries of functions that are shared between applications and utilities. The most frequently required shared library is the C runtime library (GLIBC)—around 4 Mbytes. Luckily, libraries can often be reduced in size by removing all debugging information. Running the strip command on the C runtime library brings the size down to a mere 1 Mbyte or so. However, if this is still too large for your embedded system, it is possible to turn to slightly less standardized embedded libraries. [1]

Availability of the File System:

An embedded Linux file system, unlike desktop or server implementations, must offer user-independent support for recovery in the event of power failures. Also, power consumption, size and failure rate considerations mean that the file system is likely to be running from some variant of flash or read-only memory, rather than a hard disc or other rotating media. There are a number of alternatives when using flash-based storage media. Some flash devices connect to the standard hard drive connector (IDE port) and emulate a hard disc drive. These are extremely easy to use-requiring no changes to the Linux kernel—but are generally susceptible to corruption.[1]

3. Applications of Embedded Linux

3.1 Linux in an Embedded Communications Gateway

In 1996, Pacific Gas & Electric Company (PG&E) began a project to develop a commercial-grade advanced sensor to monitor electric lines. The system was simple in concept: use lots of small cheap units that hang on the electric lines, each consisting of a few sensor elements, an embedded CPU, a spread-spectrum radio and a battery. The sensor units would monitor for certain line conditions; if those conditions occurred, the sensor would conduct measurements, and then radio the information to a master' radio unit. That master unit would, in turn, forward the information through a gateway to a server on the corporate WAN [4].

Server software would analyze the arriving data and, if warning conditions existed, would alert the electrical system operators. The other software on the WAN was capable of monitoring and performing remote configuration of the wireless sensor network. This software uses a proprietary wireless networking protocol to accomplish the control and monitoring--in effect, tunneling this protocol within TCP/IP out through the gateway system and back.

The total system comprises of three major parts: the sensor network, the client software used the data collected by the sensors and a gateway to provide a reliable link. Linux was chosen as the operating system for the gateway.

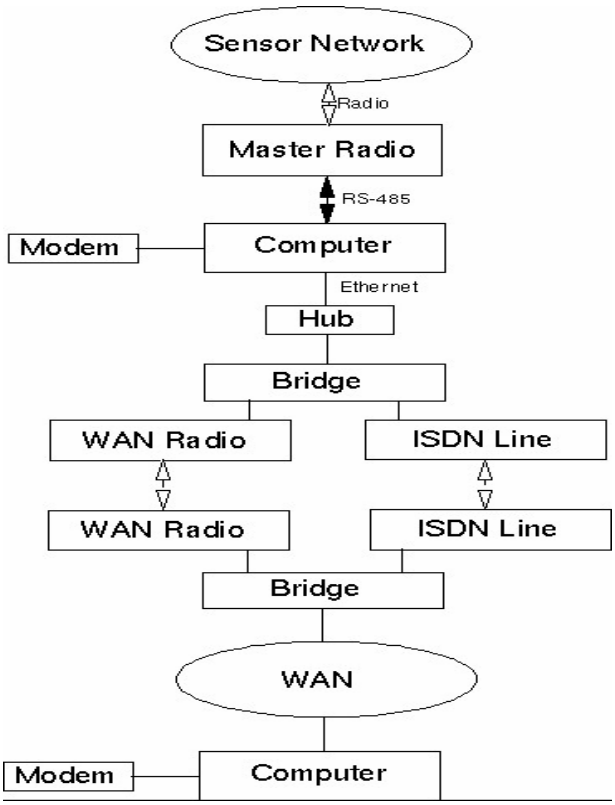


Figure 1. Gateway Schematic[4]

Reasons for choosing Linux for the?which project

Linux was immediately considered as an option for the project, due to positive experiences on an earlier project using Linux. However, the earlier project did data translations only in a batch environment in a normal office setup. While that earlier software was a major project, it was very different from a mission-essential "embedded" system. The gateway had to be remote and unattended with no recourse to human intervention if something went wrong. The designers were certain that Linux could handle the load, but alternate plans also had to be thought of. The gateway was essential to the entire project: without it, there would have been no performance metrics on how well the radio network of sensors performed. There had to be solid reasons to justify the choice of Linux for the project.

Initially there were strong proposals for using a Microsoft OS. But that approach was rejected immediately because Windows NT had some stability problems (despite being far more stable than other Windows platforms). While it is a useful desktop and/or server OS, the designers felt that NT was unsuitable for remote systems, particularly those without a display or a keyboard. Most important was the fact that the system had to operate in an environment where it had to boot and run its software automatically if the watchdog timer ever reset the system. Also the designers had little faith in NT's ability to boot, correct disk problems and then run properly in this scenario. NT and all other Microsoft Windows operating systems were rejected for these reasons. [4]

Various DOS-based solutions were also considered. There were serial port-handling libraries for DOS, which had been used successfully in the past for solving the serial communications

problems. Micro-Controller Operating System (*mu*C/OS), a freeware real-time OS available for many different microprocessors, was considered. *mu*C/OS had similarly good serial port-handling functions. Both were judged to be stable and reliable for an embedded application. However, one of the core system requirements was to allow log and data file extraction without disturbing the gateway operation. DOS would require some tricky programming to allow that option due to the uncertainty of some system calls, and *mu*C/OS would have trouble with it under high traffic loads. Portability was also another concern--the protocol tunnel code had to be easily portable to UNIX. Even if there was an affordable, reliable, TCP/IP stack for either DOS or *mu*C/OS, its ability to keep the source code portable was in serious doubts. Therefore both of those OS choices were removed from the list.

There were a lot of choices among UNIX-like operating systems: QNX, Linux, Solaris x86 or SCO. Since the designers did not have much exposure to the latter two there were doubts about their performance as none of the systems running those operating systems were particularly fast or responsive, even under fairly light loads.
So, the choices were finally reduced to QNX and Linux. But low cost, familiarity and availability of GNU tools made Linux gain the upper hand. It met all the project's key requirements at a tangible cost benefit, which was hard to ignore.

In short, Linux provided: [4]

- A stable, robust, multi-user OS
- Solid support for TCP/IP networking and serial communications
- Ability to run on various single-board and host computers
- Familiar, commonly used development tools with familiar system libraries--no time lost learning other tools and systems
- Code that was completely reusable on other UNIX platforms to implement the servers for the system
- No license fee

Results of the ? project.

The gateway system was in place for more than a year with no significant downtime and no loss of sensor data. At one point, the system had 179 days of uptime--and the only reason it needed to be rebooted was as a result of troubleshooting a problem with the network bridge equipment. Linux has proved remarkably stable and effective, and human intervention has not been needed. This project is one of many examples of Linux being successfully used in the commercial marketplace and in mission-critical commercial communication applications. In fact, Linux may have a place in such applications where perhaps no other OS can compete.[4]

3.2 The Embedded Linux “Cool Devices”

In recent months there have been numerous announcements of new Embedded Linux support for PDAs and other handheld personal computing devices. Additionally, a growing number of PDAs are known to be in development that will offer Linux as their primary embedded operating system. Some of them are listed below [5]

Sharp Zaurus SL-5500

Sharp's new PDA for the worldwide market features an embedded Linux OS supplemented by Java application environments. The device is based on a 206 MHz Intel StrongARM processor with 64MB system RAM and 16MB built-in flash storage. It has a full-color 320 x 240 pixel TFT LCD with touch panel, plus a built in keyboard and is equipped with CompactFlash and SD-card slots, and USB interfaces. The device's software stack is based on Embedix Plus PDA, which includes Lineo's Embedix embedded Linux, Trolltech's Qt/Embedded GUI application framework, Opera's Opera 5 for Linux web browser.[5]

Ericsson cordless webpad/phone

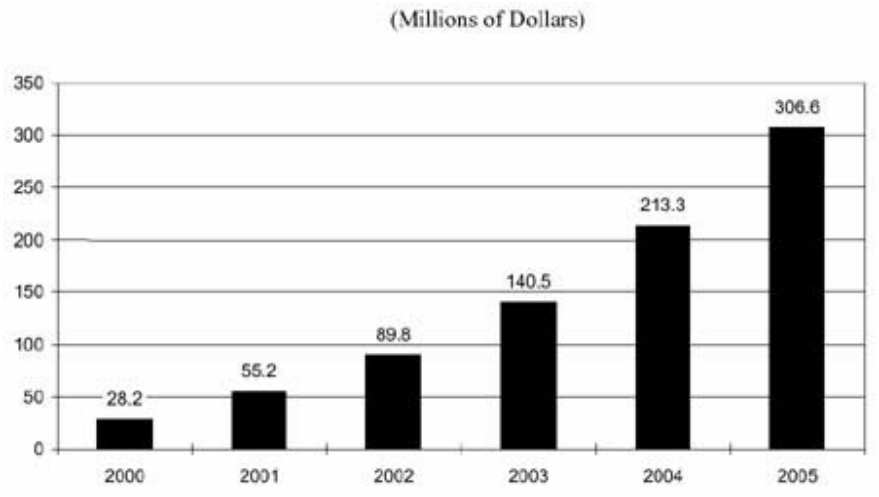
This device is basically a wireless webpad with a built-in telephone and Bluetooth wireless technology for in-home use. It can surf the web, check email, send voice clips, and make phone calls. The embedded computer is based on an Intel StrongARM processor running a Red Hat supplied Linux operating system. The GUI framework is derived from Trolltech's Qt/Embedded GUI toolkit, and the browser is from Opera.[5]

3.3 Future of Embedded Linux

The overall embedded market is undergoing a major transformation both in design and functionality. Networking technologies are becoming increasingly more important for embedded developers. Driven by the proliferation of the Internet and the increasing ubiquity of embedded computer systems, devices that can communicate with other devices are becoming dominant in the embedded market. [8]

The Linux operating system has been available for years for servers and desktops, and has continued to gain market share in both of these computing segments. Helping to drive this growth and lend creditability to Linux are well-established companies such as IBM, Dell, Hewlett-Packard, Compaq, Oracle, and others. Linux, which was originally developed for the desktop and adapted for servers, is now making way into the highly fragmented embedded market.

In 2000, worldwide shipments of embedded Linux OSs, software development tools, and related services reached an estimated $28.2 million (see figure, below). By 2005, it is estimated that shipments will reach $306.6 million, a compound annual growth rate (CAGR) of 61.2%.



Worldwide Shipments of Embedded Linux OSs, Software Development Tools, and Related Services[8]

3.4 Embedded Linux drawbacks

Even though Embedded Linux has become a disruptive force in the market today, it still has some drawbacks, which are listed below [2]

- Lack of drivers for widely varied hardware.
- Lack of standards—window managers, GUIs and real time extensions.
- Confusion over GPL license issues.
- Extreme flexibility of Linux, which could result in tradeoffs that could degrade performance.

3.5 Embedded Linux Distribution Approaches

The choice of distributions is fairly wide, with commercial offerings from Lineo, Lynuxworks and Montavista, and non-commercial variants like microLinux and the Linux router project. Individual distributions tend to add value through enhancing development tools and support for additional software components such as embedded GUIs, real time operations, etc. the non-commercial distributions are usually targeted at specific application area like embedded Internet router. The problem with these is that the support provided may not be adequate.
The second approach may be to employ an embedded Linux vendor to port their distribution to the chosen platform. Vendors such as Red Hat, Montavista and Lynuxworks port all their products to the required platforms. [1]
The third approach may be is to select hardware from a vendor who supports an embedded Linux out of the box. Arcom Control Systems provides an embedded Linux variant ported to both 486 and 586 equivalent platforms. This allows the user to concentrate on adding value through application development rather than devote time to the internals of system software.
Whatever approach is adopted, embedded Linux has now evolved to a point where it can address, at low or zero cost, and all but the most demanding embedded applications.

Give more emphasis to research issues not marketing and others?
Include a cririca overview of Linux – 1 to 2 pages
No references from technical magazines and journal papers?

3.5 Conclusion

Using Linux for an embedded system is possible and has been done. It is reliable and works.
Embedding Linux, primarily an activity of innovative software developers short while ago, has become the central focus of a rapidly growing number of commercial endeavors. The Embedded Linux Consortium, which did not exist previously has more than 75 corporate members. Major investments in embedded Linux, totaling in the hundreds of millions of dollars, have been made by industry powerhouses like Motorola, IBM, and Intel [9].
In short, embedded Linux arrived on a scene that was already in the midst of great upheaval, with developers working hard and fast to apply newly available technologies to newly defined challenges [9]. Because it provided low cost and open source, Linux spread like a wild fire in the embedded market. The open availability of source, coupled with today's ease and speed of collaboration and communication, compelled developers to quickly and efficiently adapt to the challenges of a rapidly changing embedded market.

References

1. *Embedding Linux* by **Lennon, A** IEE Review, Volume: 47 Issue: 3, May 2001 Page(s): 33 –37.

2. Presentation slides on *The state of Embedded Linux* by **Rick Lehrbaum** at the Embedded Linux Expo & Conference (ELEC) held on June 27, 2001 in San Jose
   http://www.linuxdevices.com/articles/AT2113794413.html

3. *Embedding Linux in a commercial product* by **Joel R. Williams**
   Linux journal Jan 1999.

4. *Linux in an Embedded Communications Gateway* by **Greg Herlein**
   Linux journal October 1998

5. The Linux PDAs Quick Reference Guide
   http://www.linuxdevices.com/articles/AT8728350077.html

6. *Using Linux in Embedded and Real-Time Systems* by Rick Lehrbaum
   Linux journal July 2000.

7. An article on *Embedded Linux Basics* by **Rick Cook**
   http://www.linuxworld.com/linuxworld/lw-2000-05/lw-05-embedded.html

8. White paper: *Linux's Future in the Embedded Market*
   http://www.linuxdevices.com/articles/AT4705998392.html

9. White paper: *Embedded Linux-One year later*
   http://www.linuxdevices.com/articles/AT3595894022.html