

# THE CASE FOR EMBEDDED LINUX

Joel Williams  
mailto:Joel@emlinux.com

## Linux Hits Prime Time

The Linux operating system has received a lot of Press lately as an alternative to Microsoft Windows. However, it has also found a place in the network applications, acting as an office router, Internet firewall or as web and mail servers. Several Internet service providers have built their business on Linux by using it as their service engines. It is also finding its way in many other areas. For example, Digital Domain used over one hundred Linux systems to do the graphics processing for the movie "Titanic".

## The Case For Linux

Linux is an operating system that is similar to Unix, but is a completely independent product. It has been or is being ported to run on many processors and scales well from small embedded systems to the HP IA-64 64 bit architectures. It has many of the same commands and programming interfaces Unix so that Unix people can easily work with it. Several commercial application packages have announced support for Linux, including Netscape browsers, Informix databases and the Corel office suite.

In recent years, commercial Unix vendors have been focusing on larger size systems and corporate networks. As a result, the complexity of these systems has grown quite a bit. Along with this, so has the difficulty of administering them. Linux has become popular partly because of its "back to basics" approach which makes it easier to administer than most Unix systems. Likewise people who work with both Linux and Windows NT usually find Linux much easier and more flexible to install and administer.

### *License and Royalty Fees*

Most commercial operating systems, like Windows have an up front distribution fee plus a per royalty fee. By contrast, Linux is free software commonly called Open Source Software[TM]. As such you may use it and distribute it freely, as long as you adhere to the guidelines of the General Public License available at <ftp://prep.ai.mit.edu/pub/gnu/>. Custom written software that runs on Linux may be proprietary and licensed or controlled in any way. However in the spirit of Linux; software of a generic nature is encouraged to be contributed for all to use.

In addition to the basic license fee charged by commercial operating systems, there is often a steep charge for software development tools used to create and test the program. Linux provides all of the basic tools "in the box". This includes C, C++, Java, and others. Fancier tools are available for free or modest cost. Development tools are designed to support a variety of computer architectures and debugging environments.

### *Documented*

A wealth of documentation is available ranging from user friendly tutorials for novices to Unix style cryptic manual pages. Much of this material is publicly available on the Web.

You can tell Linux is hitting “prime time” because many bookstores now have a section devoted to Linux books.

### *Network Friendly*

One area with Linux really "cooks" is in the networking area. Just about every protocol and network interface has made its appearance on Linux. Its kernel handles network protocols more efficiently than standard Unix implementations and so gives better throughput for the buck. It gets some of its advantage by using fixed size buffers, rather than overhead *mbuf* or *STREAMS* buffer allocation strategies of other flavors of Unix.

### *X Generation*

When Linux is used on a PC or workstation, the preferred Graphical User Interfaces (GUI) are based on X-Windows. This is standard fare for Unix systems. In fact, X-Windows is a platform independent client and server model that lets users at one computer open windows on another computer. This lets programs and users located on various flavors of Unix and Linux interoperate seamlessly together over a network. Linux offers a variety of window managers that provide a choice of “look and feel” models. For those who are used to Windows95, there is even a lookalike interface so that you can feel right at home (even if it is “Bills” home). Unlike Microsoft Windows, Linux does not require a graphical interface. Alternatives will be discussed later.

## The Case For Embedded Linux

### *Embedded Systems Grow Up*

The computers used to control equipment, or *embedded systems*, have been around for about as long as computers themselves. In communications they were first used back the late 60's to control electro-mechanical telephone switches. In recent times these systems have grown with the computer industry to provide ever more capabilities in smaller systems. Increasingly, these embedded systems need to be connected to some sort of network, and hence require a networking stack. For simple embedded systems, this raises the complexity level, requiring more memory, interfaces, and more services of an operating system.

Basically, when a network interface is required, or any time multiple tasks need to run concurrently, an operating system is needed. Choosing an operating system has been the source of more “Holy Wars” than the Crusades. In bye-gone days, endless comparisons were made of various performance metrics such as context switch time and interrupt latency. However, many of those wars have been fought and either won or people have simply lost interest. Most operating systems do an acceptable job of handling these issues. Linux has respectable performance in these areas as well. In fact this is part of what is behind the hot networking performance.

### *The future is just one word: tools*

In the push for shorter times to market, other factors have taken center stage. Of particular interest are the tools used to build and test the programs that run on the

operating system. Once upon a time, programmers sat for half an hour watching the software download into an In Circuit Emulator in order to debug the program. Now, the “ICE” is on the network. Downloading takes a few seconds. Often, an ICE, if one is used at all, is only used to get a new microprocessor up and running. Once it is up, most of the debugging can be done through a network connection, or serial port. Others elements of the tool chain have been improved with similar gains in development time. Linux is based on the GNU tool chain. This provides a very complete and seamless cross platform development tool chain; from editor to source level debugger. Its’ C compiler is seems to have every optimization built into it and produces very efficient executable code.

### *Real time, Really?*

One issue that comes up is the need for a “real time” operating system. The definition of real time varies quite a bit. To some people, real time means responding to an event in the 1 microsecond range, to others it is 50 milliseconds. The “hardness” of real time also varies quite a bit. Some systems need hard real time response, with short deterministic response latencies to events. However, on many systems, when analyzed closely, we see a response time requirement that is actually “near real time”. For example: one system needed to respond within 3ms 90% of the time. Often the real time requirement is actually a tradeoff of time and buffer space. With memory getting cheaper, and CPUs getting faster, *near real time* is more typical need than *hard real time*. Many commercial operating systems that claim to be real time are far from hard real time. Usually, when you get into the detailed design, with these systems, there are warnings that the drivers interrupts and applications must be very carefully designed in order to meet real time requirements. For example, vxWorks (Ref: Programmers Guide 5.3.1, Section 2.5.5) recommends reserving the highest level interrupts in a microprocessor for critical functions such a motion control on a robot arm. This is good advice. However, it applies to any operating system. The time critical functions basically need to be done at interrupt level, and the priority of these interrupts and the functions that they perform need to be carefully designed.

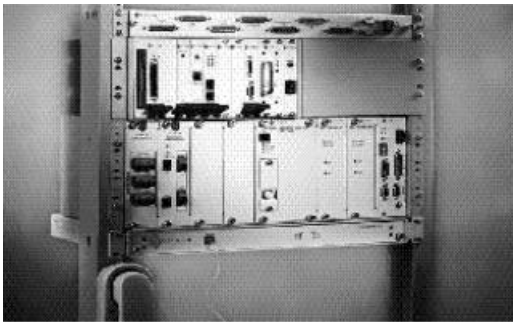
Linux, with the real time extension, (RT-Linux) provides this kind of precise control over interrupt handling. Essentially, there is an interrupt manager that handles all interrupts. It does a good job of making sure that critical interrupts get run when needed. The hardness of this approach depends mostly on the CPU interrupt structure and context switch hardware support. This approach is sufficient for a large range of real time requirements. Even without the real time extensions, Linux does pretty good at keeping up with multiple streams of events. For example, Linux PC systems running on a low end Pentium is able to keep multiple 10BaseT interfaces running without losing any packets, while running character level serial ports at a full 56KBPS, without losing any data.

### *But is Linux Embeddable?*

The typical shrink wrap Linux system has been packaged to run on a PC, with a hard disk and tons of memory. Much of this is not needed on an embedded system. A fully featured Linux Kernel requires about 1 Meg of memory. However, the Linux micro-kernel itself barely sips memory, 100K on a Pentium CPU, including virtual memory, and all core OS functions. With the networking stack and basic utilities, a

complete Linux system runs quite nicely in 500K of memory on an Intel 386 microprocessor, with an 8-bit bus (SX). The size of memory required is often dominated by the applications needed, such as a web server or SNMP agent. Embedded systems frequently do not have disk drives. Likewise, an embedded Linux system does not require one. A number of memory organizations are possible. In the simplest case Linux runs out of FLASH memory, and may use part of the flash memory as a read-only file system to store extra programs and static data.

There are many examples of embedded Linux systems. It is useful to look at just a few. The ELKS (Embeddable Linux Kernel Subset) project plans to put Linux onto a Palm Pilot. Here are two examples of currently running embedded Linux systems:

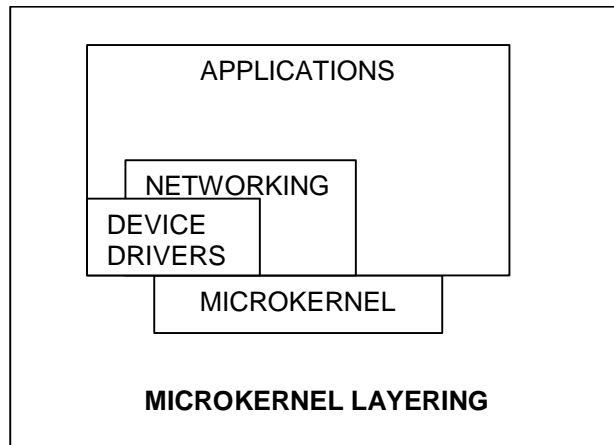


This is a commercial WDM Fiberoptic system controlled and monitored by Linux. This Linux system contains a web server, OSI and TCP/IP protocol stacks, a custom fiberoptic network interface, as well as monitoring and control interfaces. You can learn more about this system at <http://www.emlinux.com/optel.pdf>

This is a matchbook sized web server, based on an AMD 486-SX computer. You can learn more about it at: <http://wearables.stanford.edu/>



Some form of Linux can run on just about any computer that executes code. A fully featured Linux requires a 32bit internal CPU architecture.



The core operating system itself is a fairly simple micro-kernel architecture. Networking and file systems are layered on top of the micro-kernel in modular fashion. Drivers and other features can be either compiled in or added to the kernel at run time as loadable modules. This provides a highly modular building block approach to constructing a custom embeddable system. An embedded system typically uses a combination of custom drivers and application programs to provide the value added functionality. Often an embedded system also requires some generic capabilities. In order to avoid re-inventing the wheel, these are built with “off the shelf” programs and drivers. A long list of these is available for common peripherals and applications.

*It slices, it dices, it chops!*

Because Linux is scaleable and flexible, it can handle a wide range of embedded systems functions.

*User Interfaces*

If a user interface is required, a range of possibilities exists. At the low end, a custom display can be added. This can either be a direct drive LCD display requiring custom driver, or more conveniently, it may have an ASCII interface. Moving up the scale a notch is the dumb terminal serial port interface. These serial interfaces are supported by a number of “shells” or command line interpreters. A character based graphics library is available. A character based web browser (Lynx) is also available which uses this package. At the other end of the scale is a full windowing system based on X-windows. This is the powerful client-server windowing system used by all Unix Workstations. A wealth of applications run on X-Windows, including Netscape browsers. The client server model

allows users to be on a different computer than the application program; connected by a network or modem link.

The user display for graphics is usually based on a VGA interface. Monitors come in many varieties from desktop CRTs to laptop type LCD displays to the smallest display we have seen; an eye-loop display mounted on a headband. This is a tiny display that positions in front of one eye; great for wearable embedded systems.

### *Networking and Telephony*

Linux has already been used for a variety of networking and telephony applications and interfaces including:

- Internet Telephony - H.323
- Voice mail and Computer Integrated Telephony
- T1/E1 interfaces
- Network Gateway Security and Firewall services
- Web server
- Other ISP services such as a time server
- WEB Server Farms

### *Databases*

Linux supports a number of public domain databases and proprietary databases including Oracle, Sybase, Informix.

### *Languages & Interfaces*

Support for C and C++ are the most popular languages in the Linux area. A number of libraries are available to extend the capabilities of the language such as Corba object broker and DCOM interfaces.

Java Development Kit is available from Sun Microsystems.

### *Device Interfaces*

Hardware interfaces including the usual PC type interfaces, such as PCI bridges, IDE disk drives, SCSI hosts, serial and parallel ports, USB, networking interfaces of all kinds.

A long list of support for various specialized cards and chips, including flash memory, voice, video, wireless modems are also available.

### *And in conclusion...*

Linux is a versatile and cost effective operating system for embedded systems It can be embedded in a surprisingly small system to handle simple tasks and scaled up to handle more complex tasks. Linux can run on most microprocessors with a wide range of peripherals and has a ready inventory of off the shelf applications.

Development cycles are shortened through the use of mature tools, open source code, substantial documentation and available support services.