



Adoption of Linux in Embedded Devices

**Gene Sally
Product Manager**

TIMESYS CORPORATION

Linux has become the operating system of choice for embedded device projects, due to the cost savings, flexibility, and access to innovation available. This paper examines the advantages of Linux, along with the technologies that make it particularly appealing for use with embedded devices.

ADOPTION OF LINUX IN EMBEDDED DEVICES	1
THE ORIGINS OF LINUX.....	3
META TRENDS THAT ENABLED LINUX SUCCESS	3
LINUX COMES TO THE EMBEDDED MARKET.....	4
EMBEDDED LINUX TODAY	5
NETWORKING/TELECOM	5
INDUSTRIAL CONTROL	5
CONSUMER ELECTRONICS AND MOBILE DEVICES	6
STORAGE.....	6
WHY EMBEDDED LINUX?	7
COST SAVINGS	7
OPEN SOURCE	7
RAPID INNOVATION	8
TECHNOLOGY FOCUS AREAS.....	8
REAL-TIME.....	8
POWER MANAGEMENT	9
FOOTPRINT SIZE	10
OUTSOURCING LINUX, MIPS AND TIMESYS – A CASE STUDY.....	10
VALUE PROPOSITION	11
SUMMARY	12

The Origins of Linux

Linux is the namesake of a Finnish Computer Science student, Linus Torvalds, who created the Linux kernel as a non-commercial replacement for the Minix operating system. Minix was loosely based on the UNIX operating system, sharing many of its design and implementation philosophies, and was licensed to students of computer science as a learning aid for studying operating systems.

Linus Torvalds introduced Linux with this short announcement¹ on the comp.os.minix mailing list:

```

Hello everybody out there using Minix -

I'm doing a (free) operating system (just a hobby, won't be big and
professional like gnu) for 386(486) AT clones. This has been brewing
since april, and is starting to get ready. I'd like any feedback on
things people like/dislike in Minix, as my OS resembles it somewhat
(same physical layout of the file-system (due to practical reasons)
among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to
work. This implies that I'll get something practical within a few
months, and I'd like to know what features most people would want. Any
suggestions are welcome, but I won't promise I'll implement them :-)
```

Linus (torvalds@kruuna.helsinki.fi)

```

PS. Yes - it's free of any Minix code, and it has a multi-threaded
fs. It is NOT portable (uses 386 task switching etc), and it probably
never will support anything other than AT-hard disks, as that's all I
have :-).
```

The initial Linux kernel, like Minix, only ran on the Intel x86 processor and provided just the bare services offered by an operating system. The code that ran in “userspace,” such as the command line and affiliated utilities, remained the same as those shipped with Minix.

To free Linux from its Minix userspace dependencies, developers began a porting effort for code written for another operating system, HURD², which was being developed by the Free Software Foundation (FSF) as a non-commercial replacement for the UNIX operating system. Using this software from the FSF allowed the maintainers of the Linux project to offer a distribution based entirely on free software, with no licensing encumbrances from commercial entities.

The combination of the Linux kernel and the enabling “userspace” software from the FSF³ resulted in the genesis of today’s Linux operating system. Then, refinements in both the scope of userspace programs and in the kernel itself led to Linux blossoming into the operating system that it is today.

Meta Trends that Enabled Linux Success

The Linux phenomenon was enabled by several technological trends that coincided with the advent of Linux:

- 1. Widespread availability of high-speed Internet connectivity**

To create and manage a distributed development project, participants must have the ability not only to communicate well, but to be able to move around large

¹ <http://groups.google.com/group/comp.os.minix/msg/b813d52cbc5a044b>

² <http://www.gnu.org/software/hurd/history.html> – This URL provides a history of the project as well as information about its present state.

³ The complete list of projects being developed under the GNU project resides at <http://savannah.gnu.org>.

amounts of code on a routine basis. Even a small operating system like the early Linux kernel was large in an absolute sense, with contributors to it and to the FSF's userspace applications being located all over the world. The sheer amount of data generated working on such a project would be impossible to transmit without the high-speed Internet connections that are taken for granted today.

2. Standards-based software adoption

The rise of the Internet with open protocols also meant that enterprise level customers were seeking the tools and software that implemented these protocols as more customers and partners were using the Internet for communications. Since existing enterprise networking companies were slow to adopt standards-based software to connect to the Internet, enterprise customers looked to the open source community to fill this gap.

3. The GNU project

The Free Software Foundation's fostering of a set of free tools and applications by the GNU's Not Unix⁴ project enabled Linux kernel development in two key areas: by supplying the utilities that run in userspace on top of Linux, and more importantly, by providing the development tools so that users could compile Linux on their own. The success of Linux in embedded software depended greatly on the GNU development tools in that they could be easily configured to do the esoteric function of cross-compilation, a key job for embedded development.

4. Increasing software licensing costs

As hardware prices declined through the late 1980s and early 1990s, software prices remained stubbornly high, with the decline in hardware prices making the stagnant software pricing seem all the more lofty. Concurrently, Linux was gaining the features and stability so that it could be worthwhile substitute for software that supported enterprise network.

Companies Using Embedded Linux

Consumer-grade Routers

3Com
ASUS
D-Link
LinkSys
NetGear
U.S. Robotics

Mobile Devices

Haier
Motorola
NEC
Panasonic
Samsung

Consumer Electronics

D-Link
Glyph
Haier
Hauppauge
Monster Cables
Samsung
Seagate

Storage

HP
IBM
LinkSys
Panasas

Linux Comes to the Embedded Market

Most embedded systems contained software that was not built on top of an operating system; instead, the software ran directly on the hardware. Having no operating system meant that the user could create systems that had complete control over every bit of the hardware resources. The control came at the expense of having to do all of the development for the target device, which resulted in a large amount of code being written for interacting with the device at a low level, often painstakingly written in assembler code. This code didn't directly address the business problem for which the device was being developed and frequently was not usable in other projects. However,

⁴ Acronyms that refer to themselves are a running joke in the computer science field.

due to the constraints of the processors used for embedded projects, this trade-off was necessary.

In this environment, several companies developed libraries that provided basic functionality for the devices, such as task management or control of peripherals. These libraries, however, did not constitute an operating system, as the user still interacted with the processor, memory, and devices directly. These libraries provided great value, insulating the engineer from most of the intricacies of the hardware and allowing the developer to program in a higher-level language, such as C. For simple devices, using a solution like this remains a viable solution today, as some projects, due to extreme cost or power consumption constraints, cannot take advantage of an operating system.

Linux found early footing in the embedded market in the telecom space. Networking companies were pressed to deliver devices with more functionality in shorter time frames, while at the same time maintaining a high level of quality. With Linux, a device manufacturer could take advantage of the networking features that already existed in Linux and simply add their domain-specific features. Since most telecom equipment was not constrained by form factor or power consumption constraints, the additional resources required by the Linux operating system did not hinder the adoption of Linux.

Embedded Linux Today

Today, Linux finds itself embedded in a wide variety of devices, ranging from mission-critical military devices⁵ to consumer electronics. The need for advanced features and greater ease of use means that companies creating embedded systems require the operating system and userspace features supplied with a Linux-based operating system.

Networking/Telecom

Edge routers and other equipment on the fringe of a larger network were formerly complex devices that required an expert to configure them and to get them running. One example of these devices is the router that many customers have in their home which connects their computer to their Internet Service Provider for high-speed Internet. These devices need the software services to provide Internet connectivity between the service provider's network and the network in the customer's home, but must still be easy for a home computer user to configure and to install.

Linux has been deployed successfully in this market segment by manufacturers such as LinkSys and 3Com, providing both complete support for complex Internet technologies and ease-of-use features. Linux fills these requirements by allowing the system designers to take advantage of the connectivity features in Linux, but also by giving them the ability to create easy-to-use, end-user applications that can be accessed from a browser.

Industrial Control

The industrial control market has embraced Linux to power devices ranging from simple measurement/sampling devices to controlling automated assembly devices. To be successful in an industrial control environment, an operating system must be reliable and open for extensive customization. Because most industrial control solutions must be

⁵ <http://www.linuxdevices.com/news/NS5782785656.html>

able to interface with non-standard devices, using Linux to create drivers for these devices is much easier than doing so with a proprietary solution.

Linux's support for high-level languages means that developers targeting industrial control devices like Programmable Logic Controllers (PLCs) no longer need to struggle with ladder logic diagrams and proprietary languages to implement the designs. Instead, control software can be written in an open language like C, or even Perl.

Consumer Electronics and Mobile Devices

Linux appears in many consumer electronic devices, from cell phones to set-top boxes, focusing on those that need connectivity to a network or need a sophisticated user interface. Linux is ideal for these applications because it offers both robust, stable networking in the kernel and several choices for an excellent graphical library. In addition to the technical aspects, Linux distributions can be reduced in size to fit into a minimal amount of memory, thereby delivering a great amount of functionality, while keeping down bill of material costs.

Most Linux design wins in Consumer Electronics are entertainment-related devices or set-top boxes, the most ubiquitous being the TiVo DVR. Linux has found its way into a number of other products, such as the home entertainment systems by Monster Cables, iRiver, and Panasonic. In all cases, product designers needed devices that were able to connect to and to communicate with other devices on a home network or the Internet, while also providing a host of other services, such as video processing, data storage, and image processing.

Today's cell phones are sophisticated devices on par with a laptop computer: a small screen, the ability to run several programs at once, and network connectivity paired frequently with a keyboard (albeit small) and pointing device. The ideal operating system for cell phones can provide the interfaces to the lower-level devices as well as the higher-level communication protocols. Linux closely fits these requirements, and is currently deployed in devices by Motorola, Nokia, Panasonic, and Samsung.

Storage

Linux serves as an ideal choice for a storage device's operating system because of its connectivity features, device support, and high reliability. Storage devices fall into two categories: consumer and enterprise. Most consumer-level storage devices allow the connection of additional drives through a USB interface and require easy-to-use network and device configuration. Also, since consumer electronics are very price sensitive, the licensing and resource advantages of Linux allow manufacturers to bring high-powered devices to the market with minimal software development costs, providing room to adjust pricing as necessary.

In the enterprise market, Linux is a natural choice to provide the stability and power required by enterprise-caliber storage devices. The stability and scalability of Linux means that a company can focus their development resources on a single operating system platform for all of their devices. The Linux kernel has been "battle tested" in some of the most demanding systems in the world and is ready for deployment in an environment that values high performance and reliability. With Linux, engineers have complete control over every line of code in the product, in addition to the freedom to modify the operating system to meet the specific goals of the market. Linux has been

selected by vendors such as Panasas⁶, IBM, and Hewlett Packard⁷ as the core of their enterprise storage solutions.

Why Embedded Linux?

Cost Savings

There are no run-time or per-unit royalties to pay for Linux; the vast majority of users would not consider paying per-unit royalties⁸. This does not mean that Linux is without cost, as many companies prefer to purchase a ready-made Linux distribution from specialized embedded Linux vendors⁹. These distributions include productivity-boosting tools and software necessary for the customer to begin developing their applications immediately.

The savings from royalty-free distributions are two-fold: most obviously, developers and manufacturers do not need to pay the OS vendor royalties for each device shipped; less apparent is that customers no longer have the administrative overhead required to figure out how much to pay. Additionally, customers aren't exposed to audit by the vendor or to other restrictions that would reduce the company's flexibility with respect to product configuration or offerings. In fact, most TimeSys customers report that the administrative savings greatly outweigh the up-front costs associated with a per-unit royalty fee structure, and they profit greatly from the increased flexibility as an additional bonus.

Open Source

Linux is open source, meaning that there are no restrictions on getting the source code for the kernel or userspace utilities; furthermore, vendors have the obligation to supply the source code for each Linux distribution, along with instructions on how to turn that source code into the code that the computer understands. This flexibility means that the end-user, not the supplier, has the control over the software and can make changes as their business needs dictate.

Many customers who are new to Linux don't understand the importance of this defining feature of the Linux operating system. Quite simply, it means that users of Linux aren't locked into a certain vendor's business model, release cycle, or support organization. Linux users can take advantage of the changes and improvements available in the open source community as they become available. They don't have to wait for a vendor who might not be interested, for financial or other reasons, in supplying them in a timely manner, or worse, who might expect them to pay a premium for speedy updates.

Development organizations often fail to take into account the indirect costs of vendor lock-in – how it impedes the productivity of their engineers and how it affects the time to market, feature richness, and overall quality of their products. These feature and quality deficits translate directly into a diminished competitive position when others in the market do not suffer from similar encumbrances.

⁶ http://www.panasas.com/corporate_overview.html

⁷ <http://www.enterprisestorageforum.com/technology/features/article.php/3502966>

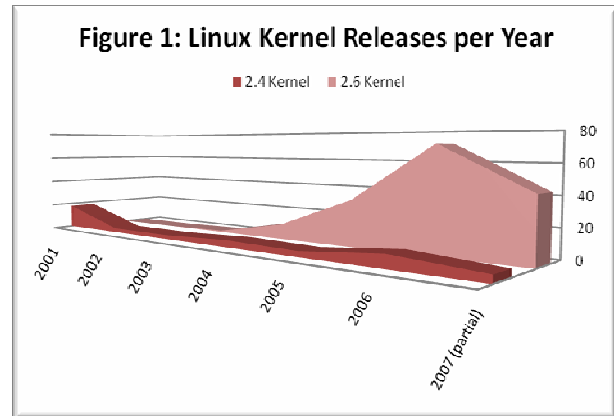
⁸ <http://www.linuxdevices.com/cgi-bin/survey/survey.cgi?view=archive&id=01302007173158> Question 12

⁹ <http://www.linuxdevices.com/cgi-bin/survey/survey.cgi?view=archive&id=01302007173158> Question 7, although the field is very fragmented

Rapid Innovation

From its start as a minimal operating system kernel whose first release cycle involved just a few engineers and took almost a year, Linux has grown into a technological juggernaut. Engineers working on the Linux kernel number in the thousands, and releases occur almost on a quarterly basis. Each release includes improvements to the core of the Linux kernel and also adds support for more peripheral devices as increased Linux adoption makes support vital for device vendors¹⁰.

The Linux Kernel Project keeps its pace of innovation through a management system by which certain people control what goes into the “mainline” kernel release by accepting changes from those working on that section. Known as “maintainers,” these people check to make sure the proposed change won’t have any ill effects on the rest of the Linux kernel before permitting the changes into the mainline. Maintainers consider and accept changes from anyone as long as the changes work; the intent is to be as open as possible to updates and changes while guarding the technical integrity of the Linux kernel. To ensure that maintainers know and understand their respective areas, they must first make substantial contributions to Linux and must also be accepted in that position by their peers.



Since Linux is more than the just the kernel, a similar process occurs for userspace programs. Most projects have a few maintainers who are typically the primary developers on the project, and who police the quality of the changes suggested by the community.

The combination of changes in the userspace and kernel means that Linux is constantly improving on all fronts at nearly all times, and at a rapid pace. Linux users have the choice to use these new improvements as they see fit or to make their own modifications when necessary.

Technology Focus Areas

Linux supplies cutting-edge technology across a wide area of different problem spaces; however, embedded Linux engineers tend to focus on a few as important in the industry. This section examines several of these key technologies, highlighting the primary benefits that Linux has to offer for each.

Real-Time

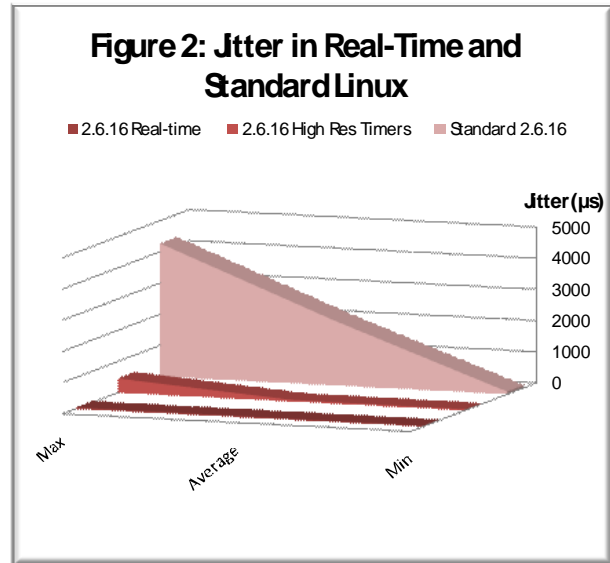
Real-time systems are those that behave in a predictable manner. The speed (or throughput) of the system, while important, isn’t the critical factor of what constitutes a real-time system. In the 2.6 version of the Linux kernel, much of the development effort is focused on creating a kernel that behaves in a predictable manner with throughput treated as an optimization. Presently, the real-time features are a separate project from

¹⁰ kernel.org, kernel change logs

the mainline Linux kernel and require some amount of customization for each target processor.

The gains in predictability using these real-time features are impressive¹¹, with an eye-popping 98% decrease in average latency, from a little over 2,000 microseconds in the standard Linux kernel to just under 35 microseconds. Since predictability is important in a real-time system, the difference between the best and worst times is important as well; in the worst case, the real-time kernel misses its deadline by 35 microseconds, or close to 50% of the average. These features make it possible to deploy Linux in a system that must meet deadlines in a predictable manner.

The figure shown to the right clearly illustrates the great improvements in predictable behavior. The flatter the line is on the graph, the more predictable the performance is. For practitioners in the real-time field, the difference between expected response time and the actual response time for a certain task is known as *jitter*. The measurements shown in the graph are the responsiveness to a timer of a system that is otherwise busy performing other tasks. The standard Linux kernel shows a jitter measurement ranging from approximately 80 microseconds to over 4,200 microseconds, an unacceptable amount of jitter for nearly any real-time application. However, the kernel with the real-time patches has a jitter of less than 45 microseconds, a difference of several orders of magnitude.



The predictable performance offered by real-time modifications to the Linux kernel make it possible for Linux to function in an environment in which it must control a machine based on data acquired from the environment. For example, if a machine sampled the height of a fluid in a container and opened an overflow valve when necessary, that machine would need to respond in a predictable manner when collecting information about the level of fluid in the reservoir, as a missed sample might cause an overflow based on the flow rate of the input. Once the fluid level reached a threshold, the system would need to open a valve within a predictable amount of time. Software that behaves in a predictable manner allows the system designers to build in the right tolerances so the fluid in the system would not reach an undesirable level.

Power Management

Power management means the efficient use of power in a system, usually implemented by the gradual shutdown of system components when they are not actively being used. This usually minimizes the amount of electricity consumed by a device, thereby extending battery life and reducing the amount of heat generated by the consumption of electricity. Linux implements power management through implementing the Advanced

¹¹ <http://rt.wiki.kernel.org/index.php/Cyclicttest> – The data in this section is from the test on a fully-loaded Intel-based processor.

Configuration and Power Interface, ACPI, an industry standard¹² used across a wide range of processors and operating systems.

ACPI provides a layer of abstraction between the underlying devices and the operating system, so that software engineers can program to a single specification and so that hardware engineers know what to build into their devices to ensure that the specification can be met. ACPI defines a number of processor and hardware states and associates a certain power state for each. Based on the system's activity, the ACPI manager switches to the appropriate state. The specification gives the hardware designer great latitude in defining the possible hardware states, while at the same time granting the user the ability to react as necessary.

The Linux ACPI implementation is actively supported, producing results similar¹³ to the implementation of power management in Microsoft's Windows XP product.

Footprint Size

Most users who are new to embedded Linux wonder how the operating system, as they experience it on their desktop, can be configured to fit on a resource-constrained device. A Linux platform (both kernel and userspace) can be whittled down to about 2MB¹⁴; a platform offering a full-featured graphical environment can be as small as 4MB.

Linux can be small for two primary reasons: first, the Linux kernel is extremely modular and can have unused bits of code removed, and second, there are many replacements for userspace parts of the platform that have been intentionally designed to be small. The majority of the code in the Linux kernel interacts with devices, while still providing basic operating system services such as memory allocation and process control. By removing the code for the devices that will never be used, the kernel's size can be greatly reduced. This is an extremely practical approach for the vast majority of embedded devices, as they have a fixed number of peripherals that never change.

Reducing the size of userspace components is accomplished by constructing the Linux platform with one of several projects which implements a minimal set of programs for the userspace. While several projects of this type exist, most embedded Linux engineers select BusyBox (www.busybox.net) because of its wide support for and tight integration with other projects specifically designed to reduce the memory footprint of a device.

Outsourcing Linux, MIPS and Timesys – A Case Study

Given the rapid growth of Linux and related projects, the challenge faced by most embedded engineering managers isn't between Linux and another operating system, but rather how to get the right Linux software platform for their project. Since all Linux projects require some degree of customization, customers require core vendors to supply ready-to-use components and tools so that subscribers can quickly customize a platform suited to their needs without a mandatory professional services engagement. MIPS Technologies satisfies this requirement through its relationship with Timesys.

¹² <http://www.acpi.info/adopt.htm> contains a list of the project adopters.

¹³ <http://www.linux-pm.org/docs/pm-summit-0406-acpi.pdf>

¹⁴ http://linuxlink.timesys.com/webinars/making_linux_work_in_a_tiny_flash

Value Proposition

The Timesys LinuxLink service offers embedded Linux engineering teams:

1. A Working Reference Distribution for a Target Processor

Getting all of the tools in place for embedded Linux development can be complex, risky, and time consuming. Subscribing to Timesys LinuxLink advances the starting point of a project by months by delivering a set of tools and components that are ready to use so that work on the value-added portion of the project can begin as soon as possible.

2. A Stable, Uniform Toolset and Environment

For organizations with several concurrent development projects, the ability to standardize on a single set of components leads to increased productivity and quality, because teams aren't solving the same problems independently. Timesys LinuxLink supplies the same versions of software components across all of its supported processors.

3. No Vendor Lock-In

Timesys LinuxLink was designed to liberate customers from vendor lock-in, even in the Linux space. Almost all of the components in Timesys LinuxLink are open source and are designed knowing that subscribers will create their own distribution with code from Timesys, from the open source community, and from themselves. Unfortunately, a small number of drivers supplied by hardware partners remain proprietary and can, therefore, only be provided in binary format, but Timesys LinuxLink delivers pre-built versions of these for all relevant platforms. Timesys supports the developers so they can be successful, no matter the origin of their software.

4. Keeping Current

After building a Linux platform, resources must be dedicated to tracking and keeping up-to-date with the constant flow of improvements and changes that occur in the open source community. The benefit of the open source community's rapid development cycle has a cost in tracking that development and deciding if new changes are appropriate for the current project. LinuxLink tracks the open community and keeps subscribers aware of pertinent changes.

5. Help and Advice

Timesys has years of embedded Linux success that help subscribers to be more productive. Timesys LinuxLink's support model puts subscribers directly in contact with other engineers who can provide immediate help without the delay of wading through a formal technical support escalation process. Support responses on Timesys LinuxLink forums can always be searched, and doing so often yields an existing answer for a similar problem.

Summary

With its feature-rich environment, wide processor coverage, and constant technological advancements, Linux is enjoying steady growth¹⁵ as an embedded operating system. Successfully using Linux for an upcoming project involves getting the right kernel and components and assembling them into a Linux platform for a specific device. While the open source community has everything necessary to assemble most platforms, engineers and their managers realize the cost and associated risk of undertaking this in a vacuum. Using Timesys LinuxLink can mitigate these concerns.

The Timesys LinuxLink subscription delivered via the web is unique in the market. Timesys LinuxLink provides not only a working distribution, but also a complete set of tools that enable subscribers to customize Linux for their particular application without cumbersome licensing or professional services engagements.

¹⁵ <http://www.embedded.com/showArticle.jhtml?articleID=192501355>